Trends in software reuse research

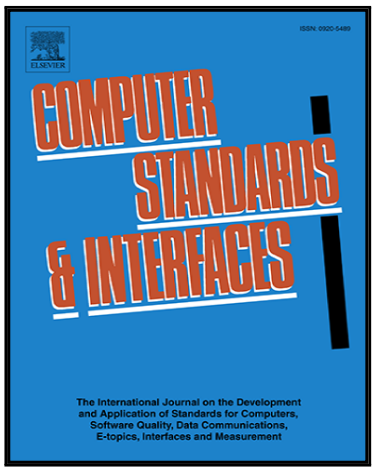
Barros-Justo, Jose L.; Benitti, Fabianne B.V.; Matalonga, Santiago

Published in:
Computer Standards & Interfaces

DOI:
10.1016/j.csi.2019.04.011

Published: 31/10/2019

Document Version
Peer reviewed version



Trends in Software Reuse Research: A Tertiary Study

José L. Barros-Justo , Fabiane B.V. Benitti , Santiago Matalonga

PII: S0920-5489(18)30463-X
DOI: https://doi.org/10.1016/j.csi.2019.04.011
Reference: CSI 3352

To appear in: *Computer Standards & Interfaces*

Received date: 19 December 2018
Revised date: 26 April 2019
Accepted date: 26 April 2019

Please cite this article as: José L. Barros-Justo , Fabiane B.V. Benitti , Santiago Matalonga , Trends in Software Reuse Research: A Tertiary Study, *Computer Standards & Interfaces* (2019), doi: https://doi.org/10.1016/j.csi.2019.04.011

Trends in Software Reuse Research: A Tertiary Study

José L. Barros-Justo[1], Fabiane B.V. Benitti[2], Santiago Matalonga[3]

[1]*School of Informatics (ESEI), Universidade de Vigo, Ourense 32004, Spain*
[2]*Universidade Federal de Santa Catarina (UFSC), Florianópolis, Santa Catarina, Brazil*
[3]*School of Computing, Engineering and Physical Science, University of the West of Scotland, Paisley PA12BE, United Kingdom*

Abstract:
Context: The reuse of software has been a research topic for more than 50 years. Throughout that time, many approaches, tools and proposed techniques have reached maturity. However, it is not yet a widespread practice and some issues need to be further investigated. The latest study on software reuse trends dates back to 2005 and we think that it should be updated.
Objective: To identify the current trends in software reuse research.
Method: A tertiary study based on systematic secondary studies published up to July 2018.
Results: We identified 4,423 works related to software reuse, from which 3,102 were filtered by selection criteria and quality assessment to produce a final set of 56 relevant studies. We identified 30 current research topics and 127 proposals for future work, grouped into three broad categories: Software Product Lines, Other reuse approaches and General reuse topics.
Conclusions: Frequently reported topics include: Requirements and Testing in the category of Lifecycle phases for Software Product Lines, and Systematic reuse for decision making in the category of General Reuse. The most mentioned future work proposals were Requirements, and Evolution and Variability management for Software Product Lines, and Systematic reuse for decision making. The identified trends, based on future work proposals, demonstrate that software reuse is still an interesting area for research. Researchers can use these trends as a guide to lead their future projects.

Keywords:
Software reuse; trends in software reuse; systematic literature review; tertiary study.

1. Introduction

The term software reuse was born 50 years ago, at the first International Conference on Software Engineering (ICSE) in 1968 [1]. Since then, researchers and practitioners have looked into software reuse for increases in productivity and cost savings. The increasing number of publications along the years show that software reuse continues to be a topic of interest in the software engineering agenda.
Throughout these years, several definitions have been proposed for software reuse. For the purposes of this paper, we will use the definition provided by the IEEE Standard for Information Technology-System and Software Life Cycle Processes-Reuse Processes [2]:

"Software reuse entails capitalizing on existing software and systems to create new products."

Under the umbrella of this broad definition, several terms coexist, such as *Component-Based Development* (CBD), *Software Product Lines* (SPL), *Model-Driven Development* (MDD), *Domain Engineering* (or *Domain Analysis*) and *Commercial-Of-The-Shelf* (COTS) among others [3]. There are also two main approaches to development: *with-reuse* (development using pre-existing components) and *for-reuse* (development of reusable components) [4]. Reuse can also be explained from the point of view of its application scope as: *vertical-reuse* (reuse of software in a given application domain) or *horizontal-reuse* (reuse of components across several application domains) [5].

Furthermore, software reuse can describe both the use of pre-existing components to develop new systems or maintaining an old product (legacy systems). Updated systems can be seen as new versions of prior *used* systems.

Literature about software reuse is abundant and includes topics such as methodologies, components, processes, economics, risks, models and organizational issues, among others. A large amount of this literature includes suggestions for future work and proposals for researchers. We are interested in identifying what these suggestions and proposals are, and how they have evolved over time.

We believe that the field needs a compass because it is dispersed in multiple solution proposals, including new techniques, tools and approaches. On the other hand, this study is necessary because there has not been, as far as we know, a revision of the state of the art for some time (the last being the work of Frakes and Kang in 2005 [6]).

To gather the evidence needed to achieve this, a systematic mapping study or a systematic literature review (secondary studies), or still better, a systematic tertiary study putting together all available evidence, systematically extracted and assessed by previous researchers [7] would be the best approach.

The main contributions of this study are:

- C1: identify how many systematic secondary studies had been published about software reuse since the inception of this term in the field of software engineering (i.e. 1968),
- C2: identify the key research topics and how they have evolved over time,
- C3: a taxonomy based on the classification of reported software reuse topics, and
- C4: a classification of proposals for future work, highlighting open research opportunities.

The rest of the paper is structured as follows: Section 2 presents the related work (i.e. other tertiary studies) in the area of interest; Section 3 describes the research method and key activities of the research protocol; Section 4 shows the Results and discuss the main findings. Section 5 reports on Validity threats. Finally, Section 6 presents the conclusions.

2. Related work

We have found only one tertiary study in the area of software reuse, specifically in the context of SPL. The work from Marimuthu and Chandrasekaran was presented at the Software Product Line Conference (SPLC'17) in September 2017 [7]. Their work aimed to identify secondary studies on SPL related topics. The authors analysed 60 secondary studies

published until December 31st 2016 and extracted data to answer five research questions. Their interests were focused on the identification of the research topics, type of secondary studies, active researchers and publication venues.

The search strategy followed by the authors consisted of the application of snowballing as the first strategy, and automatic search to complement the results. However, the procedure for the selection of seed papers is not detailed in their work, and only a reference is made to the relevance of the studies (but not to how this relevance was measured), and the number of citations reported by Google Scholar. They did not perform any additional manual search on related conferences or workshops.

They use keywording of abstracts [8] to classify the selected studies into three facets: study type, research topics and publication venues. For the research topic's facet, they reported a list of 32 items but did not explain how these topics were identified or why were they chosen.

An important outcome of the work from Marimuthu and Chandrasekaran is that the quality of the analysed secondary studies was low. As an example, out of the 60 studies analysed, only 49 reported the inclusion and exclusion criteria, only 17 applied a search strategy combining automatic search, snowballing and manual search, and, finally, the quality of the selected primary studies was explicitly analysed in only 11 studies. These numbers show a clear threat to the validity of the secondary studies included.

Marimuthu and Chandrasekaran concluded that SPL is a mature research area because the secondary studies covered a wide range of topics, although many of these studies failed to assess the quality of the primary works.

Since we only found one related tertiary study, we decided to complement this section with four other documents to provide a broader base on the topic of interest: software reuse trends.

The work from Prieto-Díaz [9] identified three key trends in software reuse in 1991, limited to the region of United States of America: 1) institutionalize the practice (i.e. make the reuse practice more "systematic"); 2) smoothly integrate reuse in the process of software development; and 3) standardize the methods for domain analysis (identification of reuse opportunities).

In 1992, Krueger [10] produced a thorough analysis of the software reuse area. He agreed with Prieto-Díaz on the need to institutionalize the practice, and the difficulties faced by industries to adopt and integrate reuse in their current software development methodologies. The main focus of the study was the identification of approaches to software reuse. To that end, they produced a taxonomy to describe and compare different approaches, by characterizing them in terms of its reusable artefacts and the way they are "*abstracted, selected, specialized and integrated*" [10]. Krueger concluded that the most crucial future work would probably be the search for high-level abstractions for software artefacts.

Zand and Samadzadeh [11] published an opinion paper in 1995. They reviewed the status and trends of software reuse by splitting the field in five key areas: 1) Organizational and management issues; 2) Business modelling and Domain analysis; 3) Technology infrastructure; 4) Storage and retrieval issues and 5) Measurement of reuse. Their final conclusion regarding the trends was that this technology could not be applied successfully unless the non-technical aspects were carefully studied, and this requires a significant extension of the perspective in reuse research.

The most recent report is from Frakes and Kang [6] in 2005. They briefly summarize the research in software reuse, including the main contributions and unresolved problems. A brief survey was passed to attendees at the Eight International Conference on Software Reuse (ICSR8). One of the four questions asked to the attendees was What are the top three remaining problems for reuse research? The most frequent answers included a variety of topics such as: scalability of solutions to very large systems, bring reuse to a broader range of software domains, better representation mechanisms for software assets, sustaining reuse programs on a long-term basis, relationship of reuse and domain engineering to newer software development processes, and the identification and validation of measures of reusability.

Our research extends the scope of Marimuthu and Chandrasekaran's work by considering the entire spectrum of software reuse approaches, not just SPL. In addition, it updates the works reported above, dating back more than 13 years. The objective of this study focuses, solely, on the identification of proposals for future work and trends in research.

3. Research method

This research is grounded on the Evidence-Based Software Engineering (EBSE) paradigm: *"EBSE (evidence-based software engineering) is concerned with determining what really works, when and where, in terms of software engineering practice, tools and standards"* [12]. The two key tools of EBSE are systematic mapping studies (SMS), also known as scope studies [8] and systematic literature reviews (SLR) [13]. The main goal of a SMS is to provide an overview of a research area, while an SLR focuses on aggregating all empirical studies on a particular topic to synthesize new knowledge.

A tertiary study is defined as "*a secondary study that uses the outputs of secondary studies as its inputs, perhaps by examining the secondary studies performed in a complete discipline or a part of it"* [14]. As observed by [14] the purpose of a secondary study is to categorise available knowledge and observe trends in the available evidence.

After 50 years of research in software reuse, we claim that the state of the field has matured enough for researchers to conduct tertiary studies. We have validated this claim by performing several pilot searches to make sure that the number and quality of available secondary studies would be suitable for the execution of a tertiary study.

The key activities to conduct a tertiary SMS are defined in a research protocol, which is detailed in the following sections.

3.1. Goal and research questions

The Goal of this study is to conduct a tertiary study to identify the trends in software reuse research and proposals for future work. To achieve this goal a tertiary SMS was conducted.

We are interested in studying how research in software reuse has evolved, and gain insight about which of these research topics are considered the current *Hot issues* in software reuse? To this end, we establish the following set of research questions (RQs):

- RQ1: How many secondary studies (SLR or SMS) were published since the inception of software reuse to date (July 2018)?

- RQ2: What research topics are being addressed in software reuse?
  - RQ2.1: How have research topics evolved over time?
- RQ3: Which proposals for future research has been reported?
  - RQ3.1: How have research topics evolved over time?

The rationale behind this set of RQs comes from the recognition that systematic reviews (SMS and SLR) are a research method aimed at identifying the available evidence in a specific field of knowledge, classifying existing data and synthesizing new knowledge. In this sense, knowing the number of secondary studies (RQ1) provides information about the interest in the research area and the amount of available evidence. Therefore, RQ1 fulfils the objective of ensuring that the available data are sufficient to guarantee valid results for the rest of the RQs.
On the other hand, RQ2 and RQ3 aim to identify current research (RQ2) and opportunities for future research (RQ3). Finally, RQ2 and RQ3 investigate evolution over time, with the aim of identifying trends.

3.2. Search strategies

Following the guidelines in [14] and [15] we designed and ran three complementary search strategies to ensure that we find the largest number of available evidence: (1) Automatic search, (2) Manual search and (3) Snowballing. To avoid bias in the automated search strategy we included four complementary electronic data sources (EDS) which are well-known among researchers and academics: ACM Digital Library (ACM DL), IEEE Xplore, SCOPUS and Web of Science (WoS), as suggested in [16]. The first two covered the most important journals and conferences in the field of software engineering [17,18] while the last two are recognized as the largest general indexing services, including papers published by ACM, IEEE, Elsevier, Springer and Wiley.

A pilot search conducted in SCOPUS allowed for the extraction of key terms related to tertiary studies in software reuse. We extracted the author's keywords from the retrieved papers and selected the most frequent ones. Tuning up the set of key terms by using synonyms and connecting them with logical operators let us evolve the search string and adapted it to every EDS, as shown in Table 1.

Table 1 Search strings adapted for every EDS

EDS	Search string	Results	Duplicate
ACM DL	acmdlTitle:(+systematic literature review mapping) AND keywords.author.keyword:(reuse reusability product-line component-based)	68	2
IEEE Xplore	("Document Title":systematic) AND (("Document Title":literature) OR ("Document Title":review) OR ("Document Title":mapping)) AND (("Author Keywords":reus*) OR ("Author Keywords":product line) OR ("Author Keywords":component-based))	14	14
SCOPUS	TITLE (systematic AND (literature OR review OR mappin g)) AND (KEY (reuse OR reusability OR "product	86	1

	line" OR "component-based")) AND (LIMIT-TO (SUBJAREA,"COMP"))		
WoS	TI=(systematic AND (literature OR review OR mapping)) AND (TS=(reuse OR reusability OR "product line" OR "component-based")) AND SU=Computer Science	74	55
	Total =	242	72

Selecting SCOPUS as the main EDS, after removing duplicates, the automated search strategy retrieved 170 unique papers (242 – 72).

The manual search strategy focuses on the proceedings of the ICSR, the most relevant conference in the research area. The following search string ran in SCOPUS on 09/august/2018, retrieved all the 483 published papers from 1994 to 2018:

"CONF (international AND conference AND on AND software AND reuse)".

Finally, we conducted a forward and backward snowballing using 27 papers retrieved by the automatic search as seeds. The selection of seed papers must guarantee that a varied set of authors, affiliations, publishing years and publication venues are considered. The amount of seeds is not as important as its quality. These 27 papers were randomly selected to avoid bias but making sure that they fulfil the recommendations suggested by Wohlin [19] and Badampudi [20]. The first iteration reviewed 1,792 references and 1,009 citations, which produced a new set of 11 seed for a second iteration (624 references and 273 citations). The snowballing process took into consideration the application of both the selection criteria and the quality assessment of papers, in order to select the appropriate seeds for every iteration and reduce effort.

3.3. Selection of works

The inclusion/exclusion criteria are summarised in Table 2. All decisions about inclusion/exclusion were based on the analysis of three reviewers (the authors), working in different pairings to help minimise bias.

The selection of the studies follows a cascade-style process, selected papers by inclusion criteria I1 serves as inputs for I2. In the same way, papers not excluded by E1 serves as input for E2 and so on.

Table 2 Inclusion/Exclusion criteria

Inclusion criteria
I1: The paper is written in English. I2: The paper reports a systematic secondary study (SMS or SLR).
Exclusion criteria
E1. Duplicate reports of the same study (we consider only the most recent one). E2: Posters, summaries of articles (less than 4 pages), books, dissertations, tutorials, slides, panels and any piece of work that can be considered as grey literature. E3: The focus of the paper is not software reuse or it did not report on future work.

Any secondary study not excluded by the above criteria will be included in the set of selected papers. The application of the exclusion criteria was done at two different levels:

1. By reviewing the meta-data information (title-abstract-keywords), if this information was not enough to exclude a paper then,
2. Review the full text, particularly the Conclusions and/or Future work sections.

Two authors (second and third), independently, carried out the process of paper selection. These authors produced two sets of pre-selected papers. The first author integrated the two previous sets, checked for inconsistencies and, when necessary, dealt with disagreements.

To deal with disagreements we applied the inclusive criteria A+B+C+D proposed in [21]. All identified papers were selected, except those papers for which both reviewers agreed to exclude them or, one author vote for exclusion while the other is in doubt.

This selection process produces a set of 74 unique papers. Unfortunately, one of these papers could not be accessed online. Seventy-three papers were used as the input for the quality assessment process described in the next section

3.4. Quality assessment

We used quality assessment criteria for filtering papers (selection process) and not for synthesis purposes. This assessment might be useful for researchers who are interested in the overall quality of the sources (secondary studies) that we used in our study, and therefore, in the validity of the extracted data.
Each secondary study (SLR or SMS) was evaluated using the DARE criteria [22]. The criteria are based on six quality assessment questions (QAs):

QA1. Are the Goal and the research questions clearly defined?
QA2. Is the literature search likely to have covered all relevant studies?
QA3. Are the review's inclusion and exclusion criteria described and appropriate?
QA4. Were the basic data/studies adequately described?
QA5. Did the reviewers assess the quality/validity of the included studies?
QA6. Are validity threats reported and appropriate?

All of the secondary studies were scored based on how well they satisfied the quality criteria described in Table 3. The overall quality score was calculated by summing up the six individual criteria scores (QA1 to QA6) as suggested in [22]. Thus, the total quality score for each study ranged between 0 (very poor) and 6 (very good). A paper has to get at least a 3.5 score (just over half the maximum score) to be included in our final set of selected studies.

Table 3 Quality ranking criteria

	Yes (1.0 score)	Partial (0.5 score)	No (0 score)
QA1	The goal and the research questions are clearly defined and reported in the study.	The goal of the study or the research questions are weakly defined.	The goal of the study or the research questions are not reported.

QA2	The authors have either searched 4 or more digital libraries and included additional search strategies or identified and referenced all journals addressing the topic of interest.	The authors have searched 2 or 3 digital libraries with no extra search strategies, or searched a defined but restricted set of journals and conference proceedings.	The authors have search up to 2 digital libraries or an extremely restricted set of journals.
QA3	The inclusion criteria are explicitly defined in the study	The inclusion criteria are implicit	The inclusion criteria are not defined and cannot be easily inferred
QA4	Information is presented about each primary study	Only summary information about primary studies is presented	The results of the primary studies are not specified
QA5	The authors have explicitly defined quality criteria and assessed every primary study	Quality issues are included as research questions	No explicit quality assessment of individual primary studies has been attempted.
QA6	Validity threats are analysed and reported;	Weak mentions to some validity threats	No mentions to validity threats.

The quality scores of all the secondary studies identified by the search strategies are reported and available online[1]. Figure 1 shows the distribution of quality scores in the set of pre-selected secondary studies from the automatic search. Almost half of the studies (25 out of 54) have a quality score between 5.0 and 6.0 (very good).

Figure 1 Distribution of quality scores

[1] https://goo.gl/SaNoyE

The first author coordinated the quality evaluation process, and allocated the set of papers to the second and third authors to do the quality assessment independently. When there was a disagreement, the authors discussed the issues until they reached an agreement.
Finally, the only pre-selected paper from the manual search strategy was excluded during the QA process.
The output of the quality assessment from the automatic search strategy produced a set of 48 selected papers. Another 11 papers were added from the snowballing process. The QA process avoided the loss of time and effort in the processing of studies that, due to their low quality, could negatively affect our results and conclusions.

3.5.Extraction of data

We created a Data Extraction Form (DEF) to objectivize the data extraction process. The DEF was implemented in a spreadsheet format with rows and columns. Columns store the data needed to answer every research question of our study, while rows represent the reviewed studies. Every cell contains a text extracted from the original source that provides data to answer the specific research question (RQ) in that column. Table 4 shows the fields that make up our DEF, their concern and the correspondence with the research questions.

Table 4 Data extraction form

#	Field	Concern/Research Question
F1	PaperID	Internal use (Identification)
F2	Author(s)	Documentation
F3	Title	Documentation
F4	Type of the publishing venue (Conference, Workshop or Journal)	Documentation
F5	Name of the publishing venue	Documentation
F6	First author's affiliation	Documentation
F7	Research topic	RQ2
F8	Number of primary papers	RQ2
F9	Proposals for Future research	RQ3
F10	Publishing Year	RQ1, RQ2.1 and RQ3.1

To reduce potential bias, we followed a procedure similar to that of the selection process. Two authors, independently, extracted data from half of the set of selected studies, while the first author did the same in a random sample that included 15 studies from each half. Further disagreements, if any, were resolved with a two-round discussion process.
Finally, three secondary studies were excluded during the process of data extraction, after noticing that they do not fully conform to the inclusion criteria. The final set of selected studies was made up of 56 works [51,23,25–50,24,52–78] that are listed in Appendix I: Reviews included in this tertiary study.
Figure 2 shows the flow and results from the three search strategies and the application of the selection criteria.

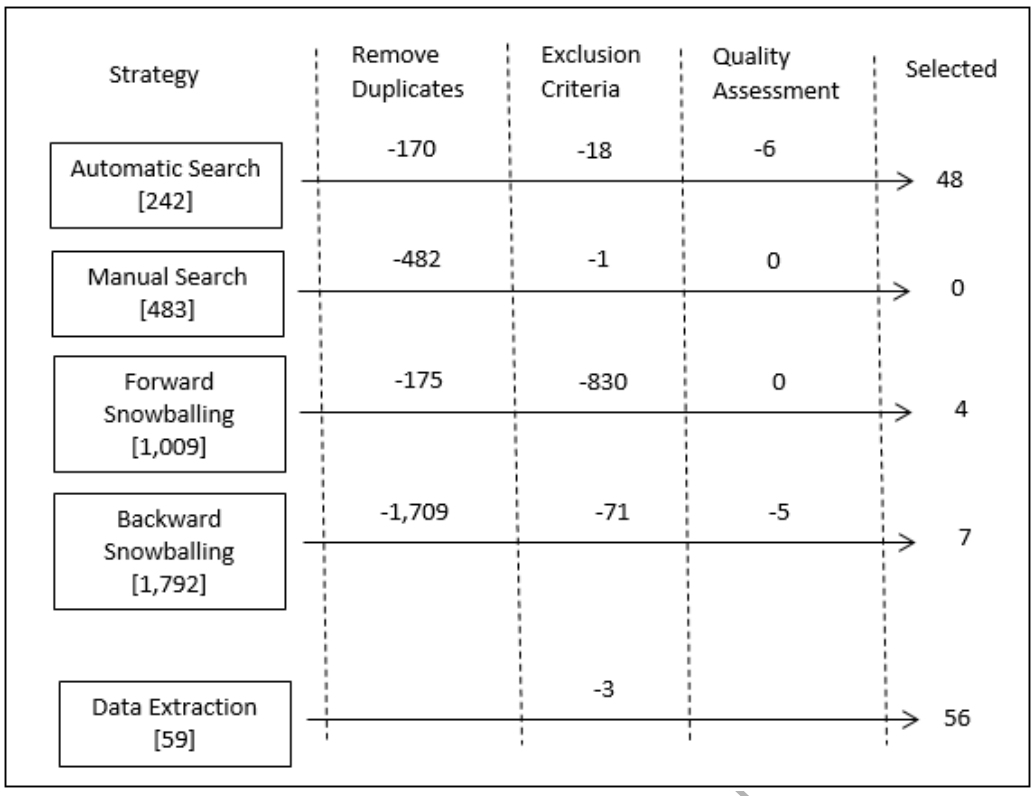


Figure 2 Final results from the search, selection, quality assessment and data extraction processes

4. Results and discussion

Throughout this work, we have followed the recommendations about good reporting practice offered by Budgen et al. in [22].

The space restrictions in a printed publication prevent the presentation of all the necessary data to ensure the total replicability of the study. For this reason, we provided an online resource to interested researchers, with all the data extracted and analysed during this work (https://goo.gl/SaNoyE). The following subsections offer a summary of the most important data, as well as a discussion of the most relevant aspects in relation to each RQ of the study.

4.1. RQ1: How many secondary studies (SLR or SMS) were published since the inception of software reuse to date (July 2018)?

Figure 3 shows the distribution of the 56 selected secondary studies published from 2007 to July 2018. Including the year 2011, and the period between 2013 and 2016, the average of published works was above eight studies per year. It was foreseeable not to find secondary studies published before 2007 since in 2004 Kitchenham published her seminal work on secondary studies [79] in the area of software engineering, but only in 2007 the guidelines for carrying them out appeared [80], and systematic secondary studies became popular in software engineering.

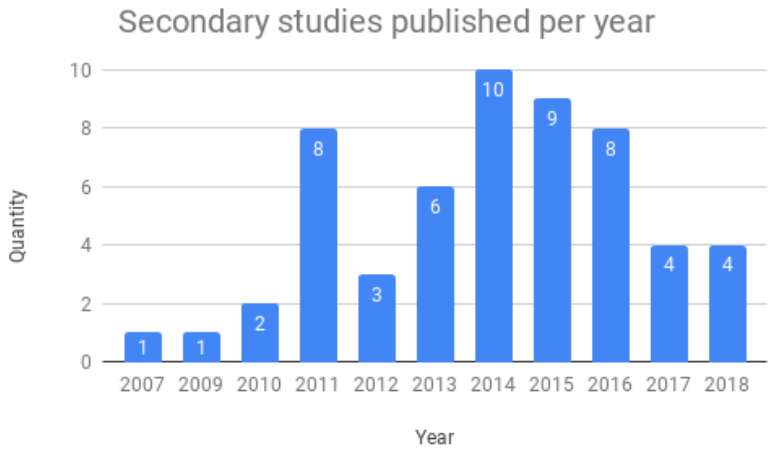


Figure 3 Distribution of secondary studies per year

4.2.RQ2: What research topics are being addressed in software reuse?

To consolidate the extraction of the research topics, we applied the Crawford Slip Method (CSM) [81]. CSM is a team classification method that relies on the use of post-it notes for classifying and grouping terms. In CSM each term to be classified is written in a post-it note, then the terms are read one-by-one. Whenever a new term is read, the classifiers can decide to add it to a previous term (thereby creating a stack) or separate it from the rest. A name must be assigned to every stack (the names of the stacks will become the names of the categories at the end of the process).

The application of the CSM method resulted in the classification structure presented below. To convey the strength of the evidence identified in each category, we use the following notation (*t, s, [SPN+]*) next to the names in each leaf category. In this notation, *t* signifies the number of individual topics that were extracted from the sources during the data extraction phase. The value of *s* relates to the number of sources, and *SNP+* stand for the identification number of the sources (can be one or more sources), whose topics were assigned to each category. For instance, Adoption of SPL (*2, 1, [SP10]*), mean that two topics (Maturity levels and SPL, and Adoption of SPL) were extracted from one source (SP10).

1. Software Product Lines (43 studies)
   - 1.1. Lifecycle Phases
     - 1.1.1. Requirements (11,13, [SP4, SP5, SP6, SP8, SP15, SP22, SP26, SP28, SP35, SP38, SP39, SP48, SP52])
     - 1.1.2. Design (3,3, [SP28, SP35, SP39])
     - 1.1.3. Development (3,3, [SP11, SP21, SP41])
     - 1.1.4. Configuration (3,3, [SP22, SP21, SP41])
     - 1.1.5. Testing (19, 8, [SP14, SP18, SP19, SP29, SP33, SP40, SP46, SP47])
     - 1.1.6. Deployment (1,1, [SP43])
   - 1.2. Project and Process Management
     - 1.2.1. Maturity levels and SPL (2,2, [SP10, SP22])
     - 1.2.2. Adoption of SPL (2,1, [SP10])
     - 1.2.3. Agile & SPL (4, 4, [SP17, SP27, SP44, SP50])
     - 1.2.4. Systematic reuse (1,1, [SP22])
     - 1.2.5. Variability Management (7,6, [SP13, SP20, SP21, SP34, SP42, SP56])
     - 1.2.6. Quality management and measurement (2,1, [SP37])

1.2.7. Tool support (1,1, [SP40])
1.2.8. Evolution & Maintenance (7,5, [SP5, SP27, SP38, SP45, SP54])
1.3. Non-functional attributes
1.3.1. Safety Engineering (1,2, [SP11, SP51])
1.3.2. Model complexity (3,3, [SP22, SP30, SP32])
1.4. Systems of systems and SPL
1.4.1. Traceability (1,1, [SP55])
1.4.2. Dependency Management across systems (1,1, [SP22])
1.4.3. Inconsistency across SPL systems (1,1, [SP22])
1.5. Search-based software engineering and SPL
1.5.1. Techniques (1,1, [SP31])
1.5.2. Application in the software lifecycle (1,1, [SP31])
2. Other reuse approaches (3 studies)
2.1. Component-based reuse
2.1.1. Testing of component-based products (1,1, [SP49])
2.1.2. Metrics to measure the quality of CBSS and its components (1,1, [SP1])
2.1.3. Open source (2,1, [SP23])
3. General reuse topics (10 studies)
3.1. Systematic reuse decision making (6,5, [SP7, SP9, SP12, SP25, SP36])
3.2. Requirements management (3,3, [SP3, SP12, SP24])
3.3. Metrics and Measurement (1,1, [SP25])
3.4. Knowledge reuse (1,1, [SP53])
3.5. Business models (2,1, [SP23])
3.6. Reference Architectures for reuse (1,1, [SP2])

Although the meaning of many of the terms in the taxonomy is well known to researchers in the area of software engineering, we have preferred to describe them below, to avoid possible ambiguities or misinterpretations.

Software product lines (category 1) refers to "a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way.", as stated by Clements and Northrop in [82].

Lifecycle phases (category 1.1) includes the specific phases of software development involving SPLE. Requirements (category 1.1.1) is concerned with the identification an analysis of real-world goals (user requirements) or limitations (system requirements). Design (category 1.1.2) includes those activities related with the search for the best solution (the most suitable), while Development (category 1.1.3) groups the two key activities: development of core assets and the production of final software applications. Configuration (category 1.1.4) includes the activities needed to adjust a set of variables, giving them adequate values, to produce the final software artefacts. Testing (category 1.1.5) aims to examine core assets, shared by many products derived from a product line, their individual parts and the interaction among them. Finally, Deployment (category 1.1.6) encompasses all the operations to prepare a system, or a new product from a product line, for its installation in the user's system.

Project and Process Management (category 1.2) includes a variety of research topics related to the adoption of SPL (category 1.2.2) such as the use of specific tools (category 1.2.7) and techniques (categories 1.2.3 and 1.2.4). The importance of quality assessment (category 1.2.6), the impact of maturity levels (category 1.2.1) and variability management (category 1.2.5) are also considered under this category, because they influence the evolution of the SPL and the maintenance tasks (category 1.2.8).

The taxonomy includes a specific entry for Non-functional attributes (category 1.3). Safety engineering (category 1.3.1) refers to the requirements stated in the functional safety standards when safety critical products are developed in product lines. Typical SPLs involve large number of features which are combined to form a huge variety of different products, implemented using multiple and different types of software artefacts. Because of the sheer amount of information, how to deal with model complexity is a hot research topic in SPL (category 1.3.2).

When faced with the use of SPL in large interconnected systems (systems of systems), professionals must ensure traceability (category 1.4.1) from the requirements to the final artifacts. Traceability in Software Product Lines (SPL) is the ability to interrelate software engineering artifacts through required links to answer specific questions related to the families of products and underlying development processes. Dependency Management across systems (category 1.4.2) deals with managing dependencies between several product lines during the distributed derivation of products, such dependencies between product lines are both technical and organizational. Finally, Inconsistency across SPL systems (category 1.4.3) is about creating consistent configurations of interdependent product lines, an activity that is challenging due to the various possible dependencies. The related product lines may be based on different types of variability models. Changes in one model can have an impact on the configuration based on another related model. Ignoring such inter-model constraints can result in invalid product configurations.

Search-based software engineering and SPL (category 1.5): A typical SPL usually involves a large number of systems and features, a fact that makes them attractive for the application of SBSE techniques, which are able to tackle problems that involve large search spaces. Techniques (category 1.5.1): There are a vast number of SBSE techniques available in the literature and many of them can be used for different purposes in SPL. Application in the software lifecycle (category 1.5.2): SBSE has been applied throughout the entire life cycle of single systems and it is possible that it can also be applied through the entire life cycle of SPLs.

Other reuse approaches (category 2) includes only one topic, component-based software engineering (category 2.1), which has been characterized by two development processes: the development of components for reuse and the development of component-based software systems (CBSS) by integrating components that have been deployed independently. We have included three subtopics in this category: the testing of component-based products (category 2.1.1), the available metrics to measure quality in CBSS (category 2.1.2) and the use of open source (category 2.1.3) as a *component*.

Finally, general reuse topics (category 3) encompass a variety of areas that can be found embedded in other categories, because of their generic nature.

To validate the resulting classification of research topics in software reuse, we compared the previous structure to topics identified by Marimuthu and Chandrasekaran [7] and by Krueger [10]. This exercise enabled us to observe that out of the 33 classification topics in [7], our classification successfully covers 27 (82%). Furthermore, we argue that our classification provides an enhanced picture of research topics on software reuse (Marimuthu and Chandrasekaran focus was restricted to SPL). Our classification scheme has a stronger internal consistency, depicted in the tree structure resulting from the CSM method when compared to the linear structure proposed in [7]. In fact, to maintain that internal consistency, our classification includes the topic "2. Other reuse approaches", although our selection process only chose papers in the subcategory "2.1 Component-based reuse". By doing that, we maintain an open classification, which can incorporate other reuse approaches in the future, while keeping internal consistency.

On the other hand, the taxonomy proposed by Krueger [10], focuses on reusable artefacts and how they are summarized, selected, specialized and integrated. Our study focuses on the open research lines and proposals for future research, which cover both development processes and artefacts. Krueger's taxonomy included the following categories of approaches for software reuse: high-level languages, design and code scavenging, source code components, software schemas, application generators, very high-level languages, transformational systems and software architectures. The differences of approach and focus, the purpose of the taxonomy and, possibly, the time elapsed between the work of Krueger (1992) and ours (2018), explains the null coincidence between both taxonomies.

4.2.1. RQ2.1: How have research topics evolved over time?

To study the interest of the topics, we mapped the year of publication of each secondary study to the topics classification taxonomy presented in section 4.2.

As Figure 4 show, there is a steady interest in Lifecycle phases in SPL (twenty-five secondary studies were published from 2009 to 2018). Within this category, requirements engineering is the one with the most sources (13 papers). The topics with the second-longest span of interest are Testing (eight studies in six years) and Configuration (3 studies identified on a six-year span).

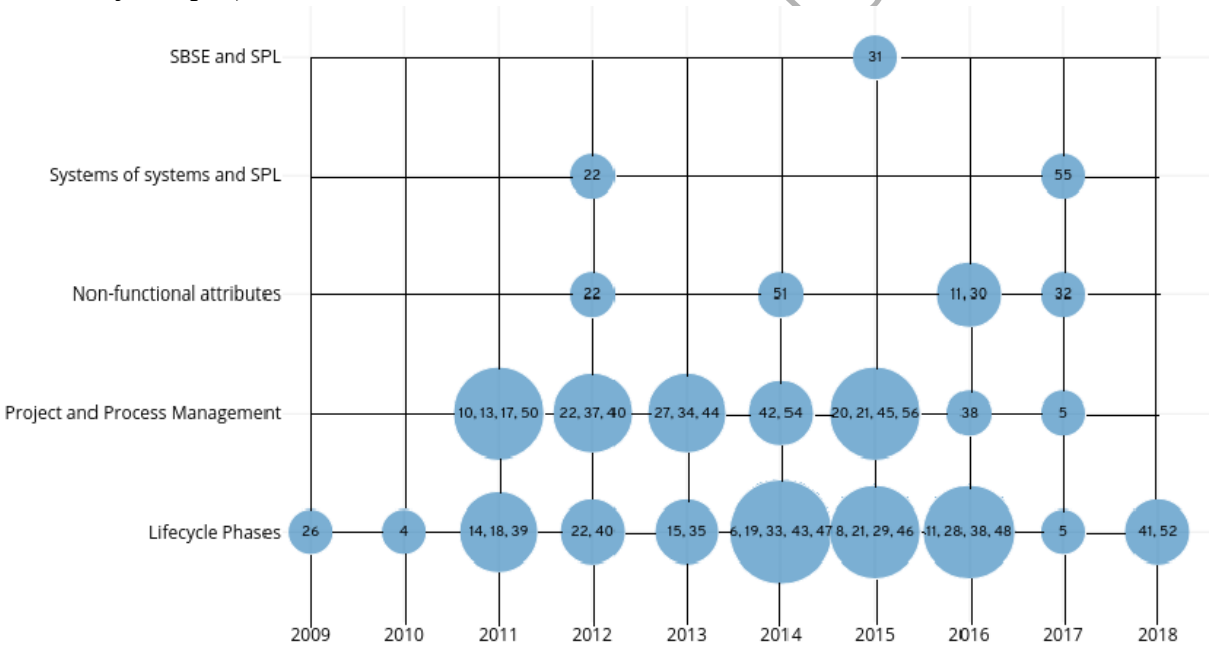


Figure 4 Trends for SPL research topics: the size of the bubble represents the number of selected papers, while the numbers within the bubble denote the bibliographic references of those papers. The Y-axis represents the main categories grouped in the SPL area (1.1 to 1.5)

Figure 5 shows that for the topics under the category General Reuse, the prevailing interest lies in the "Systematic reuse decision making". Five secondary studies [SP7, SP9, SP12, SP25, SP36] were identified under this topic and their publication spans from 2007 to 2018. The other recurring research topic is "requirements management" with three studies [SP3, SP12, SP24] published between 2014 and 2018.

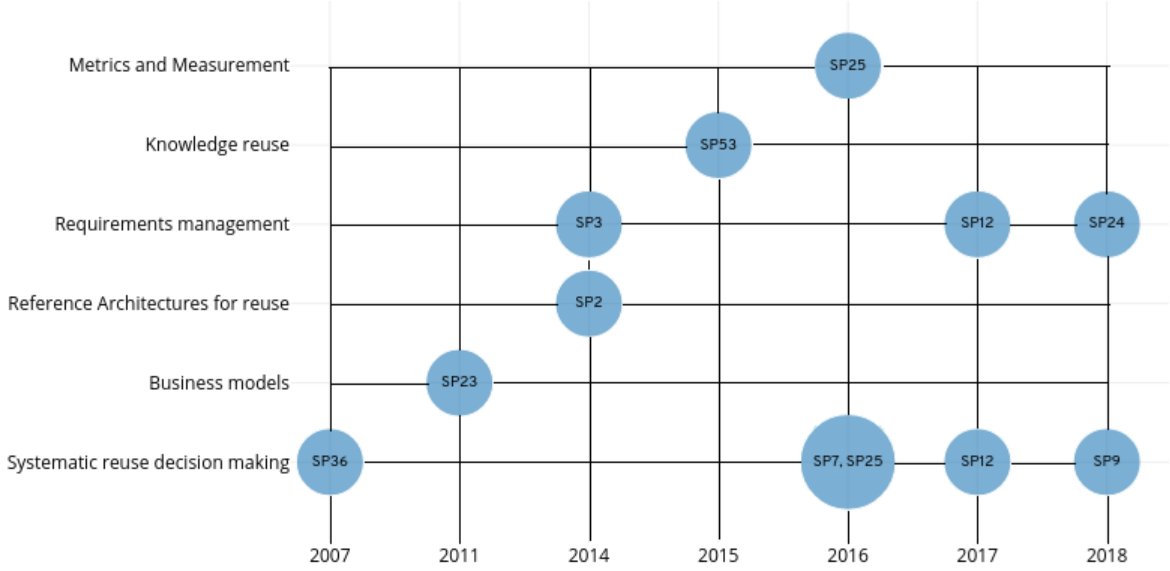
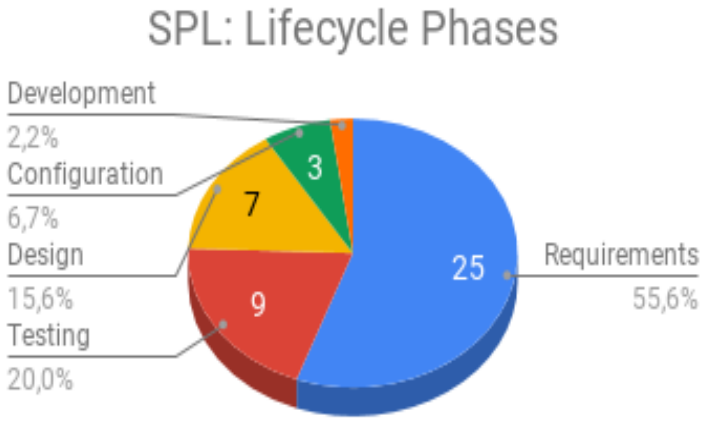


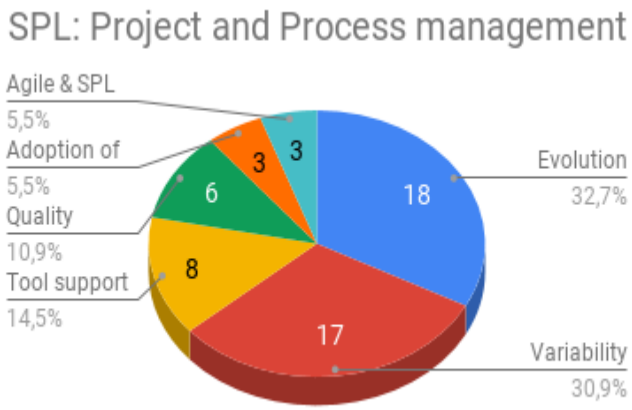
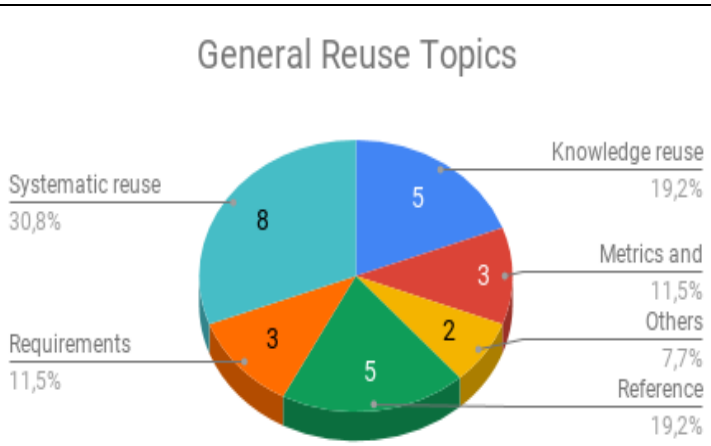
Figure 5 Trends for General reuse research topics

4.3.RQ3: Which proposals for future research has been reported?

We identified more than one hundred proposals for future research (the full list is in Appendix II: Proposals for future research). These proposals are based on the gaps identified by the authors of the 56 selected secondary studies. The gaps were identified from the reading of the full text, and were usually mentioned in the abstract, discussions or conclusions sections. In other words, the authors of the secondary studies often propose future work items that are identified gaps often from their research interest perspective. Conducting a tertiary study, we cannot identify the primary empirical evidence; we have to rely on the work from the authors of the secondary studies. In this sense, we highlight the "quality assessment" (section 3.4) that describes the quality criteria used to select these secondary studies.
Table 5 summarises the proposals for future work classified by research topic and provides a short rationale.

Table 5 proposals for future research

45 proposals. There is a clear predominance of activities related to the Requirements engineering phase, followed by Testing and Design. The proposals related to Configuration are scarce, and only one proposal about a generic Development activity was reported.	

55 proposals. The two most frequent categories are Evolution & Maintenance and Variability Management; they both reach a 64% of the identified proposals. On the other hand, the Adoption of SPL and the use of Agile development are the proposals with less frequency, which can be an indicator of the maturity of those areas.	
26 proposals. The Systematic reuse of software as a support for decision-making is the most frequent proposals. Knowledge management and the use of generic Reference architectures are the other two most reported categories. We also identified the need to develop Metrics that allow for the evaluation of performance, and to keep improving the Requirements phase to avoid misunderstandings.	

Opportunities related to SPL research area stand out, we found 116 suggestions for future work, while only 33 trends were pointed out regarding other reuse approaches or general topics about reuse.

Two points stand out as recurrent in several topics:

1. the need for more tools to support the development (SP5, SP6, SP20, SP24, SP39, SP40, SP49 and SP55), and
2. the need to run more empirical studies (SP4, SP5, SP7, SP18, SP28, SP29, SP30 and SP33), particularly involving real cases of the industry (SP6, SP26 and SP48).

4.3.1. RQ3.1: How have research topics evolved over time?

To facilitate the visualization of the trends we have divided the classification data into five graphs, three of them dedicated to the SPL category. For Figures 6 to 9, the text in the legend refers to the codes in our proposed classification scheme (i.e. Testing in Figure 6 references the topic 1.1.5 of our classification).

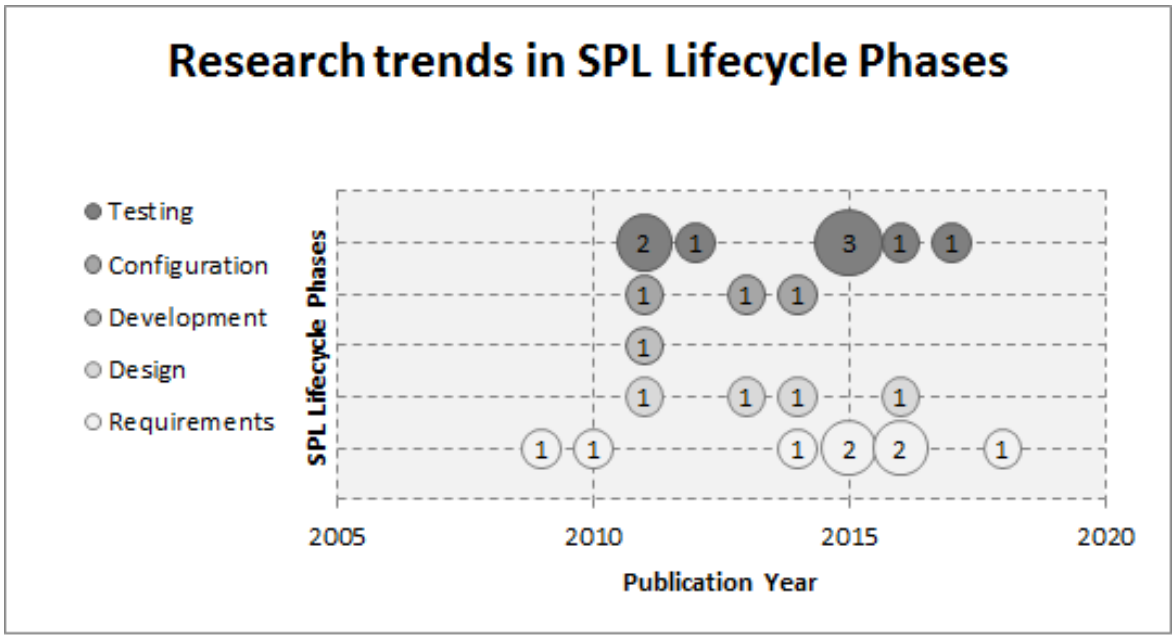


Figure 6 SPL Lifecycle Phases category: the size (and the figure inside) of the bubble represents the number of selected papers. The Y-axis represents the main categories grouped in the SPL area (1.1 to 1.5)

Our set of selected papers shows an interest in topics related to SPL Lifecycle Phases from 2009 to 2018. Data in Figure 6 demonstrate peaks on the number of publications in 2011 (5 studies), 2015 (5 studies) and 2016 (4 studies). Twenty-four studies (out of 56) were about these five topics included in the category 1.1. (SPL Lifecycle phases).

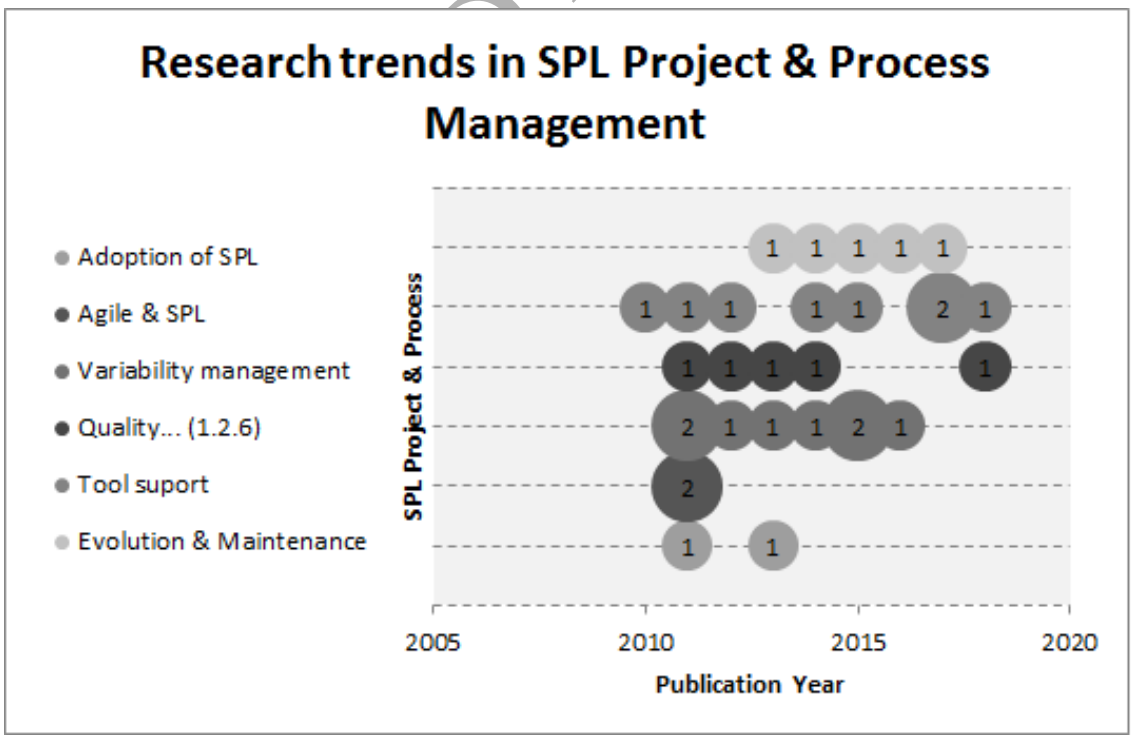


Figure 7 SPL Process & Process Management category

Thirty selected papers made proposals for future work in the area of SPL Project & Process Management. Years with more published papers were 2011 (7 studies) and 2013, 2014 and 2015 (4 studies each year). Figure 7 shows that as in category 1.1, the year 2011 was the most prolific for studies in the category 1.2.

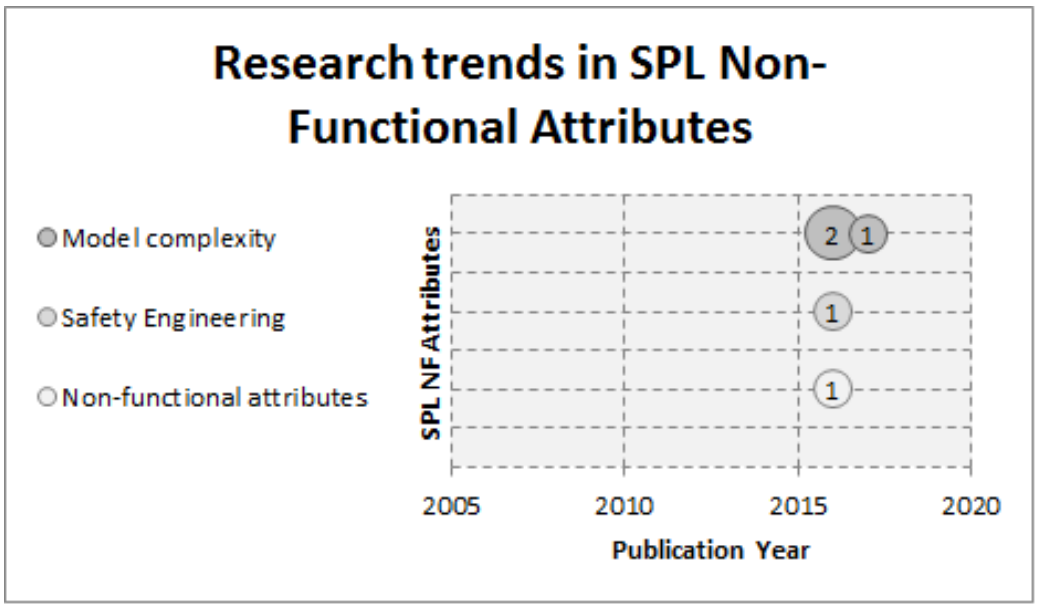


Figure 8 SPL Non-Functional Attributes category

Only five studies focus on the topic of non-functional attributes in SPL (Figure 8). Four of them were published in 2016 and one in 2017. The topics of interest include safety requirements and the management of the complexity of the models, in particular, the visualization techniques to facilitate the understanding of the features in an SPL development.

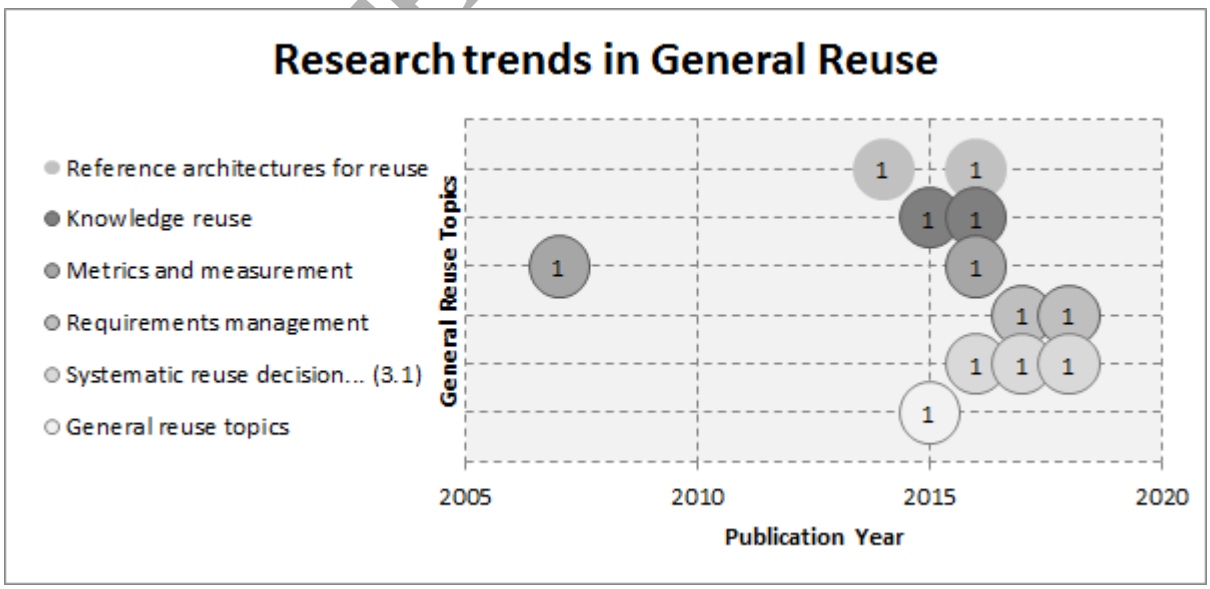


Figure 9 General Reuse category

Figure 9 shows that from the 56 studies selected, only 12 report on topics related to the category of General Reuse, not included in previous categories. Except for the SP36 study

(outsider), the other studies are concentrated in the 2014-2018 period, with four studies published in 2016. Twenty-six proposals have been extracted for future work, most of them related to the need to apply a systematic reuse approach to facilitate the decision-making process (3 papers, 8 proposals), although the topics of knowledge reuse and the use of architectures with a high level of abstraction also received great interest.
Finally, as a visual summary, Figure 10 shows a mosaic combining all the proposals for future work, classified by category and represented with distinctive colours.

Figure 10 Mosaic of Future work proposals: The left-hand side of the figure represents the categories of General Reuse (light red) and Other Reuse Approaches (Component-based reuse) in dark yellow (bottom-left). All of the others coloured rectangles represent subcategories of the SPL topic. The number after the comma indicates the amount of future work proposals drawn from each category of the taxonomy.

5. Validity threats

Being a tertiary study, the source of data for the results presented in this study all come from secondary studies. Therefore, it is possible that there are topics in software reuse research that have not yet been aggregated by secondary studies. As a result, our research methods is "blind" to those studies, and therefore they are not captured by our results. However, the number of secondary studies identified and analysed in our study (56), which in turn identified and analysed more than 2,640 primary studies, guarantees that the observed trends are significant (based on solid evidence).

A major concern in systematic secondary studies is finding all the relevant primary studies (evidence). In this case, we used three complementary search strategies to ensure that we find the largest number of related studies:

- Automatic search: we have searched in four databases covering major publications in the area of software engineering, as suggested in [5].
- Manual search: reviewing all the papers published in the ICSR (the main conference in the software reuse area, worldwide), since its first edition.
- Snowballing: backward and forward, using the guidelines from Wohlin [19].

Our search strings were designed to find the maximum number of systematic secondary studies (SMSs or SLRs), but it is possible that they missed some studies that used a different terminology to describe their review (e.g. "study aggregation" or "study synthesis"). Moreover, as there are many terms related to software reuse, our search might have been limited if we had missed a relevant synonym. This could have limited the number of retrieved papers and influenced our analysis with respect to the number of secondary studies published until today. We validated the search string by contrasting the results of our automated search with the selected papers reported by Marimuthu and Chandrasekaran [7]. The result is that out of the 60 secondary studies selected in [7]:

a) 31 papers (52%) were also selected in our study;
b) 21 papers (35%) were identified during our research process, but fail to progress to the data extraction phase, because the differences in our selection criteria; and
c) 8 papers (13%) where not identified by our search processes.

Snowballing did use only a sample (56% of the selected papers) as initial seeds, although it could have easily used all the selected papers (48) from the automated search. However, our hypothesis was that the Snowballing process achieves the same results regardless of the number of seeds used as initial set, provided that this set includes a variety of references (multiple authors, countries, organizations, years and publication venues). To this end, we repeated the SB process using all the papers selected from the results of the automatic search (48 documents) as seeds, obtaining the same results as with our original set of 27 seeds. This represents a significant saving in time and effort, given the exponential growth nature of the Snowballing process.

We did not inspect the place of the references as suggested by Wohlin but only the reference lists (backward) and the citations (forward). Inspecting the place of a reference could also suggest additional papers.

We have excluded technical reports or graduate theses retrieved by the search strategies. We assume that good quality papers based on these works would appear as journal or conference papers – particularly now that interest in systematic reviews is increasing.

The three authors agreed a preliminary review protocol, defined a data extraction form and a process to obtain consistent relevant information, and checked whether the data to be extracted would address the research questions. Moreover, as the crosscheck was necessary among the reviewers, we had at least two researchers extracting data independently. The first author dealt with any divergences and disagreements during this process.

In the process of assessing the quality of the studies, some of them did not fit perfectly into the established criteria. In this case, each evaluator defined the approximation score, and the study could be improperly excluded. To reduce this threat, the first author independently evaluated 50% of the candidate studies in a random sample and submitted the differences for discussion. A study was excluded only when the three authors reached an agreement on it.

Readers must consider that a systematic review is by definition limited by the search date, the electronic sources and the key terms used in the search. Therefore, it is possible that other papers will be included in a future replication of this study.
The authors of the secondary studies often propose future work items that are identified gaps often from their research interest perspective. These are quite subjective discussion topics or hypotheses. We summarize these proposals of future work, so our results are limited by the previous features, and by the evolution of the reuse area itself.
There is always a risk in replicating a study and find similar and consistent results, we have mitigated this validity threat by providing a detailed description of the protocol. Moreover, an additional source of complementary information is published online and publicly available.

6. Conclusions and future work

We have carried out a tertiary study with the objective of identifying trends in software reuse research. An exhaustive search in online databases, journals and conferences, allowed the identification of a significant number of secondary studies. Fifty-six studies were selected, after applying the exclusion criteria and quality assessment filters. The following conclusions are based on data extracted from those 56 studies.
The quality assessment that we performed to select the studies to be included in this review is a double-edged sword, on the one hand it can represent a validity threat (discarding studies that should be included), on the other hand it ensures that the extracted data have a strong scientific quality to justify the conclusions based on them.
As a result of the application of this research methodology and its findings, we claim the following contributions resulted from our work

- C1: An identification of the number of secondary studies published about software reuse.

The number of secondary studies published since 2007 shows an upward trend until 2014 and a slight decrease until July 2018, this is probably because of a cut-off date in mid-2018 and a delay of the search engines in indexing 2018 research. The period of greatest productivity is 2014-2016, with 27 published works (out of the 56 selected in this review). On the other hand, the number of primary studies reviewed each year exceeds 300 in 2011 and during the period 2013-2017 (our study only covers the first half of 2018).
These data confirm that interest in the research area of software reuse remains high.
While the affiliation of the first authors is very diverse, including countries in Latin America, Europe, Asia and Africa, Brazil stands out from the rest with a production of 24 studies (out of 56) in the 2010-2017 period. Also notable is the absence of studies from the USA.

- C2: The identification of key research topics and how they have evolved over time, and
- C3: Design of a classification taxonomy to classify the research topics.

Section 4.2 presents a classification taxonomy of the identified research topics in software reuse. By considering our classification taxonomy, we found that the following categories represent the prevailing trends in reuse research:

- SPL/life cycle phases/Requirements
- SPL/life cycle phases/Testing
- SPL/Management/Agile
- SPL/Management/Quality & Measurement

- SPL/Non-Functional attributes/Model complexity
- General Reuse topics/ Systematic reuse decision making

All of these categories have been mentioned (RQ2.2) for at least five years and with the last secondary study published in the last two years.

- C4. Classification of proposals for future work, highlighting open research opportunities.

If we look at future research proposals, there is a recurrent topic in a call for the development of tools for supporting the different lifecycle phases of software development. In addition to this, researchers have made evident the need for empirical studies to generate more data and knowledge about the tools and techniques that have been developed for software reuse.

When we compared the past research topics (RQ2.1) and future research proposals (RQ3.1), it is evident that the problem of incorporating reuse in the software development lifecycle is far from settled. For instance, category 1.1.1 Requirements was the focus of 13 secondary studies identified in the past nine years. Yet it is also the category with most proposals for future research. On the other hand, while Testing was the focus of 8 systematic studies, no proposal for future work was drawn from them. Apart from the interest in Requirements from both perspectives (current research and future proposals), the topics of Evolution and Variability management in SPL, and Systematic reuse for decision making in the category of General Reuse, were the most frequently mentioned proposals for future research (around 30% each one).

As future work, we intend to combine the evidence from this study with empirical data from a survey to all the authors of the selected papers. The questionnaire will focus on the current research activities and plans/expectations for future research. Once they are ranked, we think they could serve as a guide for research opportunities and identification of new tool support requirements. Another proposal for future work is the application of semiautomatic techniques such as natural language processing, neural networks, support vector machines, fuzzy logic, etc. These techniques could help create (or refine) the current taxonomy.

References

[1] M.D. McIlroy, J. Buxton, P. Naur, B. Randell, Mass-Produced Software Components, in: Proc. 1st Int. Conf. Softw. Eng., Garmisch Pattenkirchen, Germany, 1968: pp. 88–98.

[2] IEEE Standard, 1517-2010 - IEEE Standard for Information Technology--System and Software Life Cycle Processes--Reuse Processes, 2010.

[3] J. Varnell-sarjeant, A.A. Andrews, Comparing Reuse Strategies in Different Development Environments, in: Adv. Comput., 1st ed., Elsevier {BV}, 2015: pp. 1–47. doi:10.1016/bs.adcom.2014.10.002.

[4] E.S. De Almeida, A. Alvaro, D. Lucrédio, V.C. Garcia, S.R. de Lemos Meira, A survey on software reuse processes, in: Inf. Reuse Integr. Conf, 2005. IRI-2005 IEEE Int. Conf. On., 2005: pp. 66–71.

[5] M. Griss, I. Jacobson, C. Jette, B. Kessler, D. Lea, Systematic software reuse, ACM SIGSOFT Softw. Eng. Notes. 20 (1995) 17–20. doi:10.1145/223427.213969.

[6] W.B. Frakes, K. Kang, Software reuse research: Status and future, IEEE Trans. Softw. Eng. 31 (2005) 529–536.

[7] C. Marimuthu, K. Chandrasekaran, Systematic Studies in Software Product Lines: A

Tertiary Study, in: Proc. 21st Int. Syst. Softw. Prod. Line Conf. - Vol. A - SPLC '17, Seville, 2017: pp. 143–152. doi:10.1145/3106195.3106212.
[8] K. Petersen, R. Feldt, S. Mujtaba, M. Mattsson, Systematic Mapping Studies in Software Engineering, 12th Int. Conf. Eval. Assess. Softw. Eng. 17 (2007) 1–10. doi:10.1142/S0218194007003112.
[9] R. Prieto-Díaz, Software reuse trends in the United States, in: Proc. Fifteenth Annu. Int. Comput. Softw. Appl. Conf., 1991: pp. 6–7.
[10] C.W. Krueger, Software reuse, ACM Comput. Surv. 24 (1992) 131–183.
[11] M. Zand, M. Samadzadeh, Guest editors' corner Software reuse: Current status and trends, J. Syst. Softw. 30 (1995) 167–170. doi:10.1016/0164-1212(94)00131-6.
[12] EBSE Evidence-Based Software Engineering, Durham Univ. (n.d.). http://community.dur.ac.uk/ebse/ (accessed August 25, 2018).
[13] D. Budgen, P. Brereton, Performing systematic literature reviews in software engineering, Int. Conf. Soft. Engin. (2006) 1051. doi:10.1145/1134285.1134500.
[14] B.A. Kitchenham, D. Budgen, P. Brereton, Evidence-Based Software Engineering and Systematic Reviews, CRC Press, 2015.
[15] N. Bin Ali, M. Usman, Reliability of search in systematic reviews: Towards a quality assessment framework for the automated-search strategy, Inf. Softw. Technol. 99 (2018) 133–147. doi:10.1016/j.infsof.2018.02.002.
[16] N. Bin Ali, K. Petersen, Evaluating strategies for study selection in systematic literature studies, in: Proc. 8th ACM/IEEE Int. Symp. Empir. Softw. Eng. Meas., 2014: p. 45. doi:10.1145/2652524.2652557.
[17] L. Chen, M. Ali Babar, H. Zhang, Towards an evidence-based understanding of electronic data sources, in: Proc. 14th Int. Conf. Eval. Assess. Softw. Eng., 2010.
[18] M. Turner, Digital libraries and search engines for software engineering research: an overview (Technical report), 2010. https://community.dur.ac.uk/ebse/resources/notes/tools/SearchEngineIndex_v5.pdf.
[19] C. Wohlin, Guidelines for snowballing in systematic literature studies and a replication in software engineering, in: 18th Int. Conf. Eval. Assess. Softw. Eng. (EASE 2014), 2014: p. 38. doi:10.1145/2601248.2601268.
[20] D. Badampudi, C. Wohlin, K. Petersen, Experiences from using snowballing and database searches in systematic literature studies, in: Proc. 19th Int. Conf. Eval. Assess. Softw. Eng., 2015: p. 17. doi:10.1145/2745802.2745818.
[21] K. Petersen, S. Vakkalanka, L. Kuzniarz, Guidelines for conducting systematic mapping studies in software engineering: An update, Inf. Softw. Technol. 64 (2015) 1–18. doi:10.1016/j.infsof.2015.03.007.
[22] D. Budgen, P. Brereton, S. Drummond, N. Williams, Reporting systematic reviews: Some lessons from a tertiary study, Inf. Softw. Technol. 95 (2018) 62–74. doi:10.1016/j.infsof.2017.10.017.
[23] R.E. Lopez-Herrejon, S. Fischer, R. Ramler, A. Egyed, A first systematic mapping study on combinatorial interaction testing for software product lines, in: 2015 IEEE 8th Int. Conf. Softw. Testing, Verif. Valid. Work. ICSTW 2015 - Proc., 2015: pp. 1–10. doi:10.1109/ICSTW.2015.7107435.
[24] M. Abdellatief, A.B.M. Sultan, A.A.A. Ghani, M.A. Jabar, A mapping study to investigate component-based software system metrics, J. Syst. Softw. 86 (2013) 587–603. doi:10.1016/j.jss.2012.10.001.
[25] A. Ahmad, P. Jamshidi, C. Pahl, Classification and comparison of architecture evolution reuse knowledge—a systematic review, J. Softw. Evol. Process. 26 (2014) 654–691. doi:10.1002/smr.
[26] V. Alves, N. Niu, C. Alves, G. Valença, Requirements engineering for software

product lines: A systematic literature review, Inf. Softw. Technol. 52 (2010) 806–820. doi:10.1016/j.infsof.2010.03.014.

[27] W.K.G. Assunção, R.E. Lopez-Herrejon, L. Linsbauer, S.R. Vergilio, A. Egyed, Reengineering legacy applications into software product lines: a systematic mapping, Empir. Softw. Eng. 22 (2017) 2972–3016. doi:10.1007/s10664-017-9499-z.

[28] W.K.G. Assunção, S.R. Vergilio, Feature Location for Software Product Line Migration : A Mapping Study, in: Proc. 18th Int. Softw. Prod. Line Conf. Companion Vol. Work. Demonstr. Tools - SPLC '14, 2014: pp. 52–59.

[29] D. Badampudi, C. Wohlin, K. Petersen, Software component decision-making: In-house, OSS, COTS or outsourcing - A systematic literature review, J. Syst. Softw. 121 (2016) 105–124. doi:10.1016/j.jss.2016.07.027.

[30] N.H. Bakar, Z.M. Kasirun, N. Salleh, Feature extraction approaches from natural language requirements for reuse in software product lines: A systematic literature review, J. Syst. Softw. 106 (2015) 132–149. doi:10.1016/j.jss.2015.05.006.

[31] J.L. Barros-Justo, F. Pinciroli, S. Matalonga, N. Martínez-Araujo, What software reuse benefits have been transferred to the industry? A systematic mapping study, Inf. Softw. Technol. 103 (2018) 1–21. doi:10.1016/j.ijmultiphaseflow.2015.10.008.

[32] J.F. Bastos, P.A. da Mota Silveira Neto, E.S. de Almeida, S.R. de Lemos Meira, Adopting software product lines: A systematic mapping study, in: 15th Annu. Conf. Eval. Assess. Softw. Eng. (EASE 2011), 2011: pp. 11–20.

[33] S. Baumgart, J. Froberg, Functional Safety in Product Lines -- A Systematic Mapping Study, 2016 42th Euromicro Conf. Softw. Eng. Adv. Appl. (2016) 313–322. doi:10.1109/SEAA.2016.58.

[34] D. Bombonatti, M. Goulão, A. Moreira, Synergies and tradeoffs in software reuse – a systematic mapping study, Softw. - Pract. Exp. 47 (2017) 943–957. doi:10.1002/spe.2416.

[35] L. Chen, M. Ali Babar, A systematic review of evaluation of variability management approaches in software product lines, Inf. Softw. Technol. 53 (2011) 344–362. doi:10.1016/j.infsof.2010.12.006.

[36] P.A. Da Mota Silveira Neto, I. Do Carmo MacHado, J.D. McGregor, E.S. De Almeida, S.R. De Lemos Meira, A systematic mapping study of software product lines testing, Inf. Softw. Technol. 53 (2011) 407–423. doi:10.1016/j.infsof.2010.12.003.

[37] J.R.F. Da Silva, F.A.P. Da Silva, L.M. Do Nascimento, D.A.O. Martins, V.C. Garcia, The dynamic aspects of product derivation in DSPL: A systematic literature review, in: Proc. 2013 IEEE 14th Int. Conf. Inf. Reuse Integr. IEEE IRI 2013, 2013: pp. 466–473. doi:10.1109/IRI.2013.6642507.

[38] L.M.P.L.M.P. Da Silva, C.I.M.C.I.M. Bezerra, R.M.C.R.M.C. Andrade, J.M.S.J.M.S. Monteiro, Requirements Engineering and Variability Management in DSPLs Domain Engineering: A Systematic Literature Review, in: Proc. 18th Int. Conf. Enterp. Inf. Syst. (ICEIS 2016), 2016: pp. 544–551.

[39] J. Díaz, J. Pérez, P.P. Alarcón, J. Garbajosa, Agile product line engineering—a systematic literature review, Softw. - Pract. Exp. 41 (2011) 921–941. doi:10.1002/spe.

[40] E. Engström, P. Runeson, Software product line testing – A systematic mapping study, Inf. Softw. Technol. 53 (2011) 2–13. doi:10.1016/j.infsof.2010.05.011.

[41] D. Flemstrom, D. Sundmark, W. Afzal, Vertical Test Reuse for Embedded Systems: A Systematic Mapping Study, in: 2015 41st Euromicro Conf. Softw. Eng. Adv. Appl., 2015: pp. 317–324. doi:10.1109/SEAA.2015.46.

[42] I. Groher, R. Weinreich, Variability support in architecture knowledge management approaches: A systematic literature review, in: Proc. Annu. Hawaii Int. Conf. Syst. Sci., 2015: pp. 5393–5402. doi:10.1109/HICSS.2015.634.

[43] G. Guedes, C. Silva, M. Soares, J. Castro, Variability Management in Dynamic Software Product Lines: A Systematic Mapping, in: 2015 IX Brazilian Symp. Components, Archit. Reuse Softw., 2015: pp. 90–99. doi:10.1109/SBCARS.2015.20.
[44] G. Holl, P. Grünbacher, R. Rabiser, A systematic review and an expert survey on capabilities supporting multi product lines, Inf. Softw. Technol. 54 (2012) 828–852. doi:10.1016/j.infsof.2012.02.002.
[45] M. Höst, A. Oručević-Alagić, A systematic review of research on open source software in commercial software product development, Inf. Softw. Technol. 53 (2011) 616–624. doi:10.1016/j.infsof.2010.12.009.
[46] M. Irshad, K. Petersen, S. Poulding, A systematic literature review of software requirements reuse approaches, Inf. Softw. Technol. 93 (2018) 223–245. doi:10.1016/j.infsof.2017.09.009.
[47] M. Irshad, R. Torkar, K. Petersen, W. Afzal, Capturing Cost Avoidance through Reuse: Systematic Literature Review and Industrial Evaluation, in: Proc. 20th Int. Conf. Eval. Assess. Softw. Eng. - EASE '16, 2016: pp. 1–12. doi:10.1145/2915970.2915989.
[48] M. Khurum, T. Gorschek, A systematic review of domain analysis solutions for product lines, J. Syst. Softw. 82 (2009) 1982–2003. doi:10.1016/j.jss.2009.06.048.
[49] M.A. Laguna, Y. Crespo, A systematic mapping study on software product line evolution: From legacy system reengineering to product line refactoring, Sci. Comput. Program. 78 (2013) 1010–1034. doi:10.1016/j.scico.2012.05.003.
[50] C. Lima, C. Chavez, A Systematic Review on Metamodels to Support Product Line Architecture Design, in: Proc. 30th Brazilian Symp. Softw. Eng., 2016: pp. 13–22. doi:10.1145/2973839.2973842.
[51] F.J. Affonso, K.R.F. Scannavino, L.B.R. Oliveira, E.Y. Nakagawa, Reference Architectures for Self-Managed Software Systems: A Systematic Literature Review, 2014 Eighth Brazilian Symp. Softw. Components, Archit. Reuse. (2014) 21–31. doi:10.1109/SBCARS.2014.18.
[52] R.E. Lopez-Herrejon, S. Illescas, A. Egyed, Visualization for Software Product Lines: A Systematic Mapping Study, in: 2016 IEEE Work. Conf. Softw. Vis., 2016: pp. 26–35. doi:10.1109/VISSOFT.2016.11.
[53] R.E. Lopez-Herrejon, L. Linsbauer, A. Egyed, A systematic mapping study of search-based software engineering for software product lines, Inf. Softw. Technol. 61 (2015) 33–51. doi:10.1016/j.infsof.2015.01.008.
[54] R.E. Lopez-Herrejon, S. Illescas, A. Egyed, A systematic mapping study of information visualization for software product line engineering, J. Softw. Evol. Process. 30 (2018) 1–18. doi:10.1002/smr.1912.
[55] I. do C. Machado, J.D. McGregor, Y.C. Cavalcanti, E.S. de Almeida, On strategies for testing software product lines: A systematic literature review, Inf. Softw. Technol. 56 (2014) 1183–1199. doi:10.1016/j.infsof.2014.04.002.
[56] S. Mahdavi-Hezavehi, M. Galster, P. Avgeriou, Variability in quality attributes of service-based software systems: A systematic literature review, Inf. Softw. Technol. 55 (2013) 320–343. doi:10.1016/j.infsof.2012.08.010.
[57] B. Mohabbati, M. Asadi, D. Gašević, M. Hatala, H.A. Müller, Combining service-orientation and software product line engineering: A systematic mapping study, Inf. Softw. Technol. 55 (2013) 1845–1859. doi:10.1016/j.infsof.2013.05.006.
[58] P. Mohagheghi, R.R. Conradi, Quality, productivity and economic benefits of software reuse: A review of industrial studies, Empir. Softw. Eng. 12 (2007) 471–516. doi:10.1007/s10664-007-9040-x.
[59] S. Montagud, S. Abrahão, E. Insfran, S. Abrah╣o, E. Insfran, S. Abrahão, E. Insfran,

A systematic review of quality attributes and measures for software product lines, Softw. Qual. J. 20 (2012) 425–486. doi:10.1007/s11219-011-9146-7.

[60] L. Montalvillo, O. Díaz, Requirement-driven evolution in software product lines: A systematic mapping study, J. Syst. Softw. 122 (2016) 110–143. doi:10.1016/j.jss.2016.08.053.

[61] E. Murugesupillai, B. Mohabbati, D. Gaevic, A preliminary mapping study of approaches bridging software product lines and service-oriented architectures, in: Softw. Prod. Line Conf., 2011: pp. 1–8. doi:10.1145/2019136.2019149.

[62] C.R.L. Neto, P.A. da Mota Silveira Neto, E.S. de Almeida, S.R. de Lemos Meira, A Mapping Study on Software Product Lines Testing Tools, in: SEKE 2012 - Proc. 23rd Int. Conf. Softw. Eng. Knowl. Eng., 2012: pp. 628–634.

[63] L. Ochoa, O. González-Rojas, A.P. Juliana, H. Castro, G. Saake, A systematic literature review on the semi-automatic configuration of extended product lines, J. Syst. Softw. 144 (2018) 511–532. doi:10.1016/j.jss.2018.07.054.

[64] J.A.J.A. Pereira, K. Constantino, E. Figueiredo, A Systematic Literature Review of Software Product Line Management Tools, in: Int. Conf. Softw. Reuse (ICSR 2015), 2014: pp. 73–89. doi:10.1007/978-3-319-14130-5_6.

[65] P.G.G. Queiroz, R.T.V. Braga, Development of critical embedded systems using model-driven and product lines techniques: A systematic review, in: Proc. - 2014 8th Brazilian Symp. Softw. Components, Archit. Reuse, SBCARS 2014, 2014: pp. 74–83. doi:10.1109/SBCARS.2014.19.

[66] R.D.R.D. Santos Rocha, M. Fantinato, The use of software product lines for business process management: A systematic literature review, Inf. Softw. Technol. 55 (2013) 1355–1373. doi:10.1016/j.infsof.2013.02.007.

[67] A.R. Santos, R.P. de Oliveira, E.S. de Almeida, Strategies for consistency checking on software product lines, in: EASE - Int. Conf. Eval. Assess. Softw. Eng., 2015: pp. 1–14. doi:10.1145/2745802.2745806.

[68] I.S. Santos, R.M. Andrade, P.A. Santos Neto, Templates for textual use cases of software product lines: results from a systematic mapping study and a controlled experiment, J. Softw. Eng. Res. Dev. 3 (2015) 1–29. doi:10.1186/s40411-015-0020-3.

[69] I.S. Santos, R.M.C. Andrade, P.A.S. Neto, How to describe SPL variabilities in textual use cases: A systematic mapping study, in: Proc. - 2014 8th Brazilian Symp. Softw. Components, Archit. Reuse, SBCARS 2014, 2014: pp. 64–73. doi:10.1109/SBCARS.2014.16.

[70] S. Sepúlveda, A. Cravero, C. Cachero, Requirements modeling languages for software product lines: A systematic literature review, Inf. Softw. Technol. 69 (2016) 16–36. doi:10.1016/j.infsof.2015.08.007.

[71] S.P. Shashank, P. Chakka, D.V. Kumar, A systematic literature survey of integration testing in component-based software engineering, in: 2010 Int. Conf. Comput. Commun. Technol., 2010: pp. 562–568. doi:10.1109/ICCCT.2010.5640467.

[72] I.F. da Silva, P.A. da M.S. Neto, P. O'Leary, E.S. de Almeida, S.R. de L. Meira, Agile software product lines: a systematic mapping study, Softw. - Pract. Exp. 41 (2011) 899–920. doi:10.1002/spe.

[73] L.R. Soares, P. Potena, I.D.C.I. do C. Machado, I. Crnkovic, E.S. De Almeida, Analysis of Non-functional Properties in Software Product Lines: A Systematic Review, in: 2014 40th EUROMICRO Conf. Softw. Eng. Adv. Appl., Institute of Electrical & Electronics Engineers (IEEE), 2014: pp. 328–335. doi:10.1109/seaa.2014.48.

[74] L.R. Soares, P.Y. Schobbens, I. do Carmo Machado, E.S. de Almeida, Feature interaction in software product line engineering: A systematic mapping study, Inf.

Softw. Technol. 98 (2018) 44–58. doi:10.1016/j.infsof.2018.01.016.
[75] A. Souag, R. Mazo, C. Salinesi, I. Comyn-Wattiau, Reusable knowledge in security requirements engineering: a systematic mapping study, Requir. Eng. 21 (2016) 251–283. doi:10.1007/s00766-015-0220-8.
[76] G. Vale, E. Figueiredo, R. Abilio, H. Costa, Bad smells in software product lines: A systematic review, in: Proc. - 2014 8th Brazilian Symp. Softw. Components, Archit. Reuse, SBCARS 2014, 2014: pp. 84–94. doi:10.1109/SBCARS.2014.21.
[77] T. Vale, E.S. de Almeida, V. Alves, U. Kulesza, N. Niu, R. de Lima, Software product lines traceability: A systematic mapping study, Inf. Softw. Technol. 84 (2017) 1–18. doi:10.1016/j.infsof.2016.12.004.
[78] S. Younoussi, O. Roudies, All about software reusability: A systematic literature review, J. Theor. Appl. Inf. Technol. 76 (2015) 64–75.
[79] B. Kitchenham, Procedures for performing systematic reviews, Keele, UK, Keele Univ. 33 (2004) 28. doi:10.1.1.122.3308.
[80] B. Kitchenham, S. Charters, Guidelines for performing Systematic Literature Reviews in Software Engineering. Version 2.3, 2007. doi:10.1145/1134285.1134500.
[81] H.W. Dettmer, Brainpower networking using the Crawford Slip method, Trafford, 2003.
[82] P. Clements, L. Northrop, No Title, Softw. Prod. Lines Pract. Patterns. (2001).

Appendix I: Reviews included in this tertiary study

ID	Full Reference
[SP1]	M. Abdellatief, A.B.M. Sultan, A.A.A. Ghani, M.A. Jabar, A mapping study to investigate component-based software system metrics, J. Syst. Softw. 86 (2013) 587–603. doi:10.1016/j.jss.2012.10.001.
[SP2]	F.J. Affonso, K.R.F. Scannavino, L.B.R. Oliveira, E.Y. Nakagawa, Reference Architectures for Self-Managed Software Systems: A Systematic Literature Review, 2014 Eighth Brazilian Symp. Softw. Components, Archit. Reuse. (2014) 21–31. doi:10.1109/SBCARS.2014.18.
[SP3]	A. Ahmad, P. Jamshidi, C. Pahl, Classification and comparison of architecture evolution reuse knowledge—a systematic review, J. Softw. Evol. Process. 26 (2014) 654–691. doi:10.1002/smr.
[SP4]	V. Alves, N. Niu, C. Alves, G. Valença, Requirements engineering for software product lines: A systematic literature review, Inf. Softw. Technol. 52 (2010) 806–820. doi:10.1016/j.infsof.2010.03.014.
[SP5]	W.K.G. Assunção, R.E. Lopez-Herrejon, L. Linsbauer, S.R. Vergilio, A. Egyed, Reengineering legacy applications into software product lines: a systematic mapping, Empir. Softw. Eng. 22 (2017) 2972–3016. doi:10.1007/s10664-017-9499-z.
[SP6]	W.K.G. Assunção, S.R. Vergilio, Feature Location for Software Product Line Migration : A Mapping Study, in: Proc. 18th Int. Softw. Prod. Line Conf. Companion Vol. Work. Demonstr. Tools - SPLC '14, 2014: pp. 52–59.
[SP7]	D. Badampudi, C. Wohlin, K. Petersen, Software component decision-making: In-house, OSS, COTS or outsourcing - A systematic literature review, J. Syst. Softw. 121 (2016) 105–124. doi:10.1016/j.jss.2016.07.027.
[SP8]	N.H. Bakar, Z.M. Kasirun, N. Salleh, Feature extraction approaches from natural language requirements for reuse in software product lines: A systematic literature review, J. Syst. Softw. 106 (2015) 132–149. doi:10.1016/j.jss.2015.05.006.
[SP9]	J.L. Barros-Justo, F. Pinciroli, S. Matalonga, N. Martínez-Araujo, What software reuse benefits have been transferred to the industry ? A systematic mapping study, Inf. Softw. Technol. (2018).
[SP10]	J.F. Bastos, P.A. da Mota Silveira Neto, E.S. de Almeida, S.R. de Lemos Meira, Adopting software product lines: A systematic mapping study, in: 15th Annu. Conf. Eval. Assess. Softw. Eng. (EASE 2011), 2011: pp. 11–20.
[SP11]	S. Baumgart, J. Froberg, Functional Safety in Product Lines -- A Systematic Mapping Study, 2016 42th Euromicro Conf. Softw. Eng. Adv. Appl. (2016) 313–322. doi:10.1109/SEAA.2016.58.
[SP12]	D. Bombonatti, M. Goulão, A. Moreira, Synergies and tradeoffs in software

	reuse – a systematic mapping study, Softw. - Pract. Exp. 47 (2017) 943–957. doi:10.1002/spe.2416.
[SP13]	L. Chen, M. Ali Babar, A systematic review of evaluation of variability management approaches in software product lines, Inf. Softw. Technol. 53 (2011) 344–362. doi:10.1016/j.infsof.2010.12.006.
[SP14]	P.A. Da Mota Silveira Neto, I. Do Carmo MacHado, J.D. McGregor, E.S. De Almeida, S.R. De Lemos Meira, A systematic mapping study of software product lines testing, Inf. Softw. Technol. 53 (2011) 407–423. doi:10.1016/j.infsof.2010.12.003.
[SP15]	J.R.F. Da Silva, F.A.P. Da Silva, L.M. Do Nascimento, D.A.O. Martins, V.C. Garcia, The dynamic aspects of product derivation in DSPL: A systematic literature review, in: Proc. 2013 IEEE 14th Int. Conf. Inf. Reuse Integr. IEEE IRI 2013, 2013: pp. 466–473. doi:10.1109/IRI.2013.6642507.
[SP16]	L.M.P. Da Silva, C.I.M. Bezerra, R.M.C. Andrade, J.M.S. Monteiro, Requirements Engineering and Variability Management in DSPLs Domain Engineering: A Systematic Literature Review, in: Proc. 18th Int. Conf. Enterp. Inf. Syst. (ICEIS 2016), 2016: pp. 544–551.
[SP17]	J. Díaz, J. Pérez, P.P. Alarcón, J. Garbajosa, Agile product line engineering—a systematic literature review, Softw. - Pract. Exp. 41 (2011) 921–941. doi:10.1002/spe.
[SP18]	E. Engström, P. Runeson, Software product line testing – A systematic mapping study, Inf. Softw. Technol. 53 (2011) 2–13. doi:10.1016/j.infsof.2010.05.011.
[SP19]	D. Flemstrom, D. Sundmark, W. Afzal, Vertical Test Reuse for Embedded Systems: A Systematic Mapping Study, in: 2015 41st Euromicro Conf. Softw. Eng. Adv. Appl., 2015: pp. 317–324. doi:10.1109/SEAA.2015.46.
[SP20]	I. Groher, R. Weinreich, Variability support in architecture knowledge management approaches: A systematic literature review, in: Proc. Annu. Hawaii Int. Conf. Syst. Sci., 2015: pp. 5393–5402. doi:10.1109/HICSS.2015.634.
[SP21]	G. Guedes, C. Silva, M. Soares, J. Castro, Variability Management in Dynamic Software Product Lines: A Systematic Mapping, in: 2015 IX Brazilian Symp. Components, Archit. Reuse Softw., 2015: pp. 90–99. doi:10.1109/SBCARS.2015.20.
[SP22]	G. Holl, P. Grünbacher, R. Rabiser, A systematic review and an expert survey on capabilities supporting multi product lines, Inf. Softw. Technol. 54 (2012) 828–852. doi:10.1016/j.infsof.2012.02.002.
[SP23]	M. Höst, A. Oručević-Alagić, A systematic review of research on open source software in commercial software product development, Inf. Softw. Technol. 53 (2011) 616–624. doi:10.1016/j.infsof.2010.12.009.
[SP24]	M. Irshad, K. Petersen, S. Poulding, A systematic literature review of software requirements reuse approaches, Inf. Softw. Technol. 93 (2018) 223–245. doi:10.1016/j.infsof.2017.09.009.

[SP25]	M. Irshad, R. Torkar, K. Petersen, W. Afzal, Capturing Cost Avoidance through Reuse: Systematic Literature Review and Industrial Evaluation, in: Proc. 20th Int. Conf. Eval. Assess. Softw. Eng. - EASE '16, 2016: pp. 1–12. doi:10.1145/2915970.2915989.
[SP26]	M. Khurum, T. Gorschek, A systematic review of domain analysis solutions for product lines, J. Syst. Softw. 82 (2009) 1982–2003. doi:10.1016/j.jss.2009.06.048.
[SP27]	M.A. Laguna, Y. Crespo, A systematic mapping study on software product line evolution: From legacy system reengineering to product line refactoring, Sci. Comput. Program. 78 (2013) 1010–1034. doi:10.1016/j.scico.2012.05.003.
[SP28]	C. Lima, C. Chavez, A Systematic Review on Metamodels to Support Product Line Architecture Design, in: Proc. 30th Brazilian Symp. Softw. Eng., 2016: pp. 13–22. doi:10.1145/2973839.2973842.
[SP29]	R.E. Lopez-Herrejon, S. Fischer, R. Ramler, A. Egyed, A first systematic mapping study on combinatorial interaction testing for software product lines, in: 2015 IEEE 8th Int. Conf. Softw. Testing, Verif. Valid. Work. ICSTW 2015 - Proc., 2015: pp. 1–10. doi:10.1109/ICSTW.2015.7107435.
[SP30]	R.E. Lopez-Herrejon, S. Illescas, A. Egyed, Visualization for Software Product Lines: A Systematic Mapping Study, in: 2016 IEEE Work. Conf. Softw. Vis., 2016: pp. 26–35. doi:10.1109/VISSOFT.2016.11.
[SP31]	R.E. Lopez-Herrejon, L. Linsbauer, A. Egyed, A systematic mapping study of search-based software engineering for software product lines, Inf. Softw. Technol. 61 (2015) 33–51. doi:10.1016/j.infsof.2015.01.008.
[SP32]	R.E. Lopez-Herrejon, S. Illescas, A. Egyed, A systematic mapping study of information visualization for software product line engineering, J. Softw. Evol. Process. 30 (2018) 1–18. doi:10.1002/smr.1912.
[SP33]	I. do C. Machado, J.D. McGregor, Y.C. Cavalcanti, E.S. de Almeida, On strategies for testing software product lines: A systematic literature review, Inf. Softw. Technol. 56 (2014) 1183–1199. doi:10.1016/j.infsof.2014.04.002.
[SP34]	S. Mahdavi-Hezavehi, M. Galster, P. Avgeriou, Variability in quality attributes of service-based software systems: A systematic literature review, Inf. Softw. Technol. 55 (2013) 320–343. doi:10.1016/j.infsof.2012.08.010.
[SP35]	B. Mohabbati, M. Asadi, D. Gašević, M. Hatala, H.A. Müller, Combining service-orientation and software product line engineering: A systematic mapping study, Inf. Softw. Technol. 55 (2013) 1845–1859. doi:10.1016/j.infsof.2013.05.006.
[SP36]	P. Mohagheghi, R. Conradi, Quality, productivity and economic benefits of software reuse: A review of industrial studies, Empir. Softw. Eng. 12 (2007) 471–516.
[SP37]	S. Montagud, S. Abrahão, E. Insfran, S. Abrah[illegible]o, E. Insfran, S. Abrahão, E. Insfran, A systematic review of quality attributes and measures for

	software product lines, Softw. Qual. J. 20 (2012) 425–486. doi:10.1007/s11219-011-9146-7.
[SP38]	L. Montalvillo, O. Díaz, Requirement-driven evolution in software product lines: A systematic mapping study, J. Syst. Softw. 122 (2016) 110–143. doi:10.1016/j.jss.2016.08.053.
[SP39]	E. Murugesupillai, B. Mohabbati, D. Gaevic, A preliminary mapping study of approaches bridging software product lines and service-oriented architectures, in: Softw. Prod. Line Conf., 2011: pp. 1–8. doi:10.1145/2019136.2019149.
[SP40]	C.R.L. Neto, P.A. da Mota Silveira Neto, E.S. de Almeida, S.R. de Lemos Meira, A Mapping Study on Software Product Lines Testing Tools, in: SEKE 2012 - Proc. 23rd Int. Conf. Softw. Eng. Knowl. Eng., 2012: pp. 628–634.
[SP41]	L. Ochoa, O. González-Rojas, A.P. Juliana, H. Castro, G. Saake, A systematic literature review on the semi-automatic configuration of extended product lines, J. Syst. Softw. 144 (2018) 511–532. doi:10.1016/j.jss.2018.07.054.
[SP42]	J.A. Pereira, K. Constantino, E. Figueiredo, A Systematic Literature Review of Software Product Line Management Tools, in: Int. Conf. Softw. Reuse (ICSR 2015), 2014: pp. 73–89. doi:10.1007/978-3-319-14130-5_6.
[SP43]	P.G.G. Queiroz, R.T.V. Braga, Development of critical embedded systems using model-driven and product lines techniques: A systematic review, in: Proc. - 2014 8th Brazilian Symp. Softw. Components, Archit. Reuse, SBCARS 2014, 2014: pp. 74–83. doi:10.1109/SBCARS.2014.19.
[SP44]	R. Dos Santos Rocha, M. Fantinato, The use of software product lines for business process management: A systematic literature review, Inf. Softw. Technol. 55 (2013) 1355–1373. doi:10.1016/j.infsof.2013.02.007.
[SP45]	A.R. Santos, R.P. de Oliveira, E.S. de Almeida, Strategies for consistency checking on software product lines, in: EASE - Int. Conf. Eval. Assess. Softw. Eng., 2015: pp. 1–14. doi:10.1145/2745802.2745806.
[SP46]	I.S. Santos, R.M. Andrade, P.A. Santos Neto, Templates for textual use cases of software product lines: results from a systematic mapping study and a controlled experiment, J. Softw. Eng. Res. Dev. 3 (2015) 1–29. doi:10.1186/s40411-015-0020-3.
[SP47]	I.S. Santos, R.M.C. Andrade, P.A.S. Neto, How to describe SPL variabilities in textual use cases: A systematic mapping study, in: Proc. - 2014 8th Brazilian Symp. Softw. Components, Archit. Reuse, SBCARS 2014, 2014: pp. 64–73. doi:10.1109/SBCARS.2014.16.
[SP48]	S. Sepúlveda, A. Cravero, C. Cachero, Requirements modeling languages for software product lines: A systematic literature review, Inf. Softw. Technol. 69 (2016) 16–36. doi:10.1016/j.infsof.2015.08.007.
[SP49]	S.P. Shashank, P. Chakka, D.V. Kumar, A systematic literature survey of integration testing in component-based software engineering, in: 2010 Int.

	Conf. Comput. Commun. Technol., 2010: pp. 562–568. doi:10.1109/ICCCT.2010.5640467.
[SP50]	I.F. da Silva, P.A. da M.S. Neto, P. O'Leary, E.S. de Almeida, S.R. de L. Meira, Agile software product lines: a systematic mapping study, Softw. - Pract. Exp. 41 (2011) 899–920. doi:10.1002/spe.
[SP51]	L.R. Soares, P. Potena, I. do C. Machado, I. Crnkovic, E.S. de Almeida, Analysis of Non-functional Properties in Software Product Lines: A Systematic Review, in: 2014 40th EUROMICRO Conf. Softw. Eng. Adv. Appl., Institute of Electrical & Electronics Engineers (IEEE), 2014: pp. 328–335. doi:10.1109/seaa.2014.48.
[SP52]	L.R. Soares, P.Y. Schobbens, I. do Carmo Machado, E.S. de Almeida, Feature interaction in software product line engineering: A systematic mapping study, Inf. Softw. Technol. 98 (2018) 44–58. doi:10.1016/j.infsof.2018.01.016.
[SP53]	A. Souag, R. Mazo, C. Salinesi, I. Comyn-Wattiau, Reusable knowledge in security requirements engineering: a systematic mapping study, Requir. Eng. 21 (2016) 251–283. doi:10.1007/s00766-015-0220-8.
[SP54]	G. Vale, E. Figueiredo, R. Abilio, H. Costa, Bad smells in software product lines: A systematic review, in: Proc. - 2014 8th Brazilian Symp. Softw. Components, Archit. Reuse, SBCARS 2014, 2014: pp. 84–94. doi:10.1109/SBCARS.2014.21.
[SP55]	T. Vale, E.S. de Almeida, V. Alves, U. Kulesza, N. Niu, R. de Lima, Software product lines traceability: A systematic mapping study, Inf. Softw. Technol. 84 (2017) 1–18. doi:10.1016/j.infsof.2016.12.004.
[SP56]	S. Younoussi, O. Roudies, All about software reusability: A systematic literature review, J. Theor. Appl. Inf. Technol. 76 (2015) 64–75.

Appendix II: Proposals for future research

Full list of identified proposals for future work

Proposal ID	Proposal for Future Research	Research Topic	Paper ID	Publishing Year
1	Revising the existing definition of CBSS metrics for better precision in measurement.	2.1.2	SP1	2013
2	Combining more than one metric based on logical conditions by which a subset of problems is detected, to characterize and evaluate CBSS with real information.			
3	Combination of the existing design techniques and knowledge types for the design of RAs&RMs for SMSS.	3.6	SP2	2014
4	Evaluation of RAs&RMs for SMSS.			
5	Establishment of a terminology for the self-* software systems domain.			
6	Periodically updating the SLR to monitor the evolution of RAs&RMs for SMSS.			
7	Reuse knowledge mining and discovery: Knowledge capturing and identification.	3.4	SP3	2014
8	Dynamic, runtime evolution: architecture change mining as a complementary and integrated phase for architecture change execution.	3.6		
9	Enhance the use of Natural Language Processing and Information Retrieval techniques in addressing variability within the mostly textual nature of requirements.	1.1.1	SP4	2010
10	Focus not only on the proactive product line adoption strategy, but also on extractive and reactive strategies and their combinations.			
11	Conduct more comparative studies, e.g., by empirically assessing the cost-effective degrees of different methods and techniques.			
12	Build an empirical base for sharing the cross-checking data, including requirements documents, requirements models, tools, validation results, etc.			
13	Conduct and report empirical studies more rigorously.			
14	Automation and tool support.	1.2.7 1.2.8	SP5	2017
15	Exploiting multiple sources of information for reengineering.	1.2.8		
16	Feature management			

17	Hybrid approaches			
18	Refactoring techniques			
19	Need of use guidelines			
20	New measures and metrics	1.2.6		
21	More robust empirical evaluation	1.2.6		
22	The implementation of tools to perform the phases is fundamental to the practice and use in the industry.	1.1.1 1.2.7	SP6	2014
23	Empirical evaluation considering real cases.	1.1.1		
24	Automatic recovery of constraints.			
25	Hybrid approaches can improve the results when compared with only one kind of strategy.			
26	To combine different sources of information to improve the results in the feature location/mapping.			
27	That new measures and metrics to evaluate reuse opportunities of artefacts in different abstraction levels.			
28	Applying search-based algorithms			
29	The implementation of the entire process to support automatic migration of existing variants to an SPL.			
30	Determining the order of importance and magnitude of the factors.	3.1	SP7	2016
31	Providing empirical evidence on comparisons of component origins with regard to the factors.			
32	Proposal of novel solutions taking (#30) and (#31) into consideration.			
33	Suitable metrics in the context of requirements reuse for SPLE.	1.1.1	SP8	2015
34	To conduct a systematic literature review to deepen the knowledge about the processes of reuse and how the benefits are transferred to the industry.	3.1	SP9	2018
35	To investigate the development of simpler protocols that can be used in industry to gather relevant data, such as return on investment, while applying rigorous methods.			
36	Use a quality model to integrate the benefits of reuse and relate them to specific reuse processes.			
37	Develop and validate a model suitable to link reuse benefits to economic values (strategic or financial).			
38	To consider the relationships between SPL adoption and factors such as company maturity and organization structure	1.2.2	SP10	2011

	in more detail.			
39	Patterns to assist in SPL adoption and overcoming SPL adoption barriers.			
40	Case Studies in Safety Engineering and SPL.	1.3.1	SP11	2016
41	To build a catalogue for software reuse to better understand how exactly the different non-functional requirements and context factors affect reusability.	3.1 3.2	SP12	2017
42	A large majority of the reported VM approaches have not been sufficiently evaluated using scientifically rigorous methods.	1.2.5	SP13	2011
43	Development of new approaches to managing variability in increasingly large and complex family of systems.			
44	How to maintain the traceability between development and test artefacts, and the management of variability through the whole development life cycle.	1.1.6	SP14	2011
45	To use the acquired knowledge about dynamic derivation to improve a product derivation process in a context-aware DSOPL.	1.1.4	SP15	2013
46	Modelling and treatment of NFRs still in domain analysis.	1.1.1 1.2.5	SP16	2016
47	Evaluating quality attributes using assets different of variability models.			
48	A mechanism to check the consistency and that be able to modify the variability model at runtime.			
49	Defining a pattern to represent the variability in DSPLs.			
50	Exploring approaches that can be used for eliciting the FRs.			
51	Determining approaches to support the found gaps and to define a formal process for DSPLs RE and VM.			
52	APLE (Agile Product Line Engineering) architecture and APLE traceability are still open research challenges.	1.2.3	SP17	2011
53	To be able to support traces between features and core-assets to easily implement maintenance tasks in a systematic and (semi)-automatic way.			
54	Launch empirical studies that use and evaluate the proposals to give solid foundation to SPL testing.	1.1.6	SP18	2011
			SP29	2015
			SP33	2014
55	The feasibility and practical applicability of applying different approaches to vertical test reuse in embedded systems development in general, and in automotive system development in particular.	1.1.6	SP19	2014
56	Is necessary to integrate variability modelling, capturing of	1.2.7	SP20	2015

	related design decisions, and architectural solutions in one tool environment.	1.2.5		
57	There is still very weak evidence for variability in the context of SAKM (software architecture knowledge management). The published approaches have not yet been applied in case studies or industrial projects.	1.2.5		
58	Investigate interaction between Non-functional requirements and context and its effect on variability configuration.	1.2.5	SP21	2015
59	Methods are needed to support structuring large models in the context of MPL (Multi product lines).	1.2.5	SP22	2012
60	Development of methods to support sharing of models.			
61	Primary study on dependencies across SPL.			
62	How companies can transform their proprietary software to open source and build a community on it and more case studies on implementation of specific methodologies for dealing with different aspects of open source in industry.	2.1.3	SP23	2011
63	To identify the reasons behind the lack of industrial validation of requirements reuse approaches.	3.2	SP24	2018
64	Varieties of different tools are used by different requirement reuse approaches. A separate study is required to analyse the working of these tools, their strengths, and weaknesses.	1.2.7 3.2		
65	Validate reuse cost models and metrics with the industry.	3.3	SP25	2016
66	The presence of empirical evidence of any sort with at least some intent to explain the overall design and execution of a validation (e.g. a pilot test in industry).	1.1.1	SP26	2009
67	The integrated or guided reengineering of (typically object-oriented) legacy code and requirements	1.2.8	SP27	2013
68	Specific aspect-oriented or feature-oriented refactoring into SPLs.			
69	Refactoring for the evolution of existing product lines.			
70	Definition of a decision model to support the selection of the metamodel according to the SPL project.	1.1.2 1.3.2	SP28	2016
71	Investigation on the migration/transformation from one metamodel to another.			
72	Performing empirical studies to investigate the use of these metamodels in SPL projects.			
73	Experimental evaluation of the testing techniques identified.	1.1.6	SP29	2015
74	To take a closer look on the interaction capabilities on the approaches to visualization for SPL and analyse the empirical foundations on which they rely upon.	1.3.2	SP30	2016

75	The developing adequate community-wide testing benchmarks, exploiting more SPL knowledge to reduce the search effort, the strong need to provide robust tooling support.	1.5	SP31	2015
76	Product line scoping and design in the area of manufacturing and marketing that relies on SBSE techniques.			
77	To perform a comparative study of the tools that visualize feature models with special focus on scalability, and apply visualization techniques for SPL testing.	1.1.6 1.3.2	SP32	2017
78	Call for empirical assessment as means to improve accuracy of the identified testing strategies	1.1.6	SP33	2014
79	Case studies on variability quality attributes in service-based systems.	1.2.5 1.2.6	SP34	2013
80	Integration or adoption of SO (service-orientation) in SPLE (especially related to DSLPs).	1.1.2	SP35	2013
81	Verifying economic returns of reuse, using comparable and consistent metrics for measuring reuse.	3.3	SP36	2007
82	Evaluating reuse of COTS or OSS components, integrating reuse activities in software processes, better data collection and evaluating return on investment.			
83	To define a quality model for SPL.	1.2.6	SP37	2012
84	To conduct primary studies to validate measures for SPL is an opportunity			
85	Studies on FOP, AOP or DOP took the form of academic evaluations aiming at proving their resiliency upon SPL evolution. No evidences were found on the applicability of these approaches in an industrial setting.	1.2.8	SP38	2016
86	Decision-making on whether product specifics should be promoted to SPL core assets.			
87	Change impact analysis upon architectural changes.			
88	Inconsistency detection for assets other than variability models. “Document change” was left out since no study was found on this activity.			
89	Exploring better ways to tailor the service granularity of service-oriented product line to enhance reusability.	1.1.4	SP39	2011
90	To conduct extensive experimental study to measure effectiveness of approach so that it can be further explained in a service composition process.	1.2.6		
91	To take service oriented product line architecture approach and apply it in different domains to validate the real benefits.	1.1.2 1.2.6		
92	Approach to include dynamic adaptation of agents and integration to support automation.	1.1.2 1.2.7		

93	Propose to create supporting tools and models to aid in service oriented application development and variability management.	1.2.7 1.1.3 1.2.5		
94	There is no tool suitable to all testing levels of a SPL, researchers need to consider the feasibility of adapting existing tools or constructing new tools.	1.2.7 1.1.6	SP40	2012
95	More experiments involving SPL testing tools are needed.			
96	Benchmark quality attributes in semi-automatic configurations of SPL.	1.2.6	SP41	2018
97	Lack of industrial support during product configuration.	1.1.4	SP42	2014
98	The definition of process/software metrics to compare approaches about the combination of PLE (Product Line Engineering) and MDE (Model-Driven Engineering) for the development of SCES (Safety-Critical Embedded Systems), and conduct experiments to try to measure its effectiveness.	1.2.6	SP43	2014
99	PL approaches for BPM are still at an early stage and gaining maturity.	1.2.2	SP44	2013
100	Faster feedback on consistency checking	1.1.6	SP45	2015
101	A robust framework to support coevolution of the SPL artefacts would probably improve the evolution process	1.2.8		
102	Together with the handling inconsistencies may be reasonable to track findings, decisions and actions made through consistency management policies			
103	To propose a new SPL use case template or to investigate which template is better.	1.1.1	SP46	2015
104	Experimental study comparing different templates to SPL variabilities in textual use cases in terms of ease of use or comprehensibility.	1.2.5	SP47	2014
105	Research on requirements modelling languages for SPLs has generated a myriad of languages that differ in the set of constructs provided to express SPL requirements. Their general lack of empirical validation and adoption in industry, together with their differences in maturity, draws the picture of a discipline that still needs to evolve.	1.1.1	SP48	2016
106	Is CBSE suitable when there are frequently changing requirements (i.e. I Agile fashion)?	2.1.1	SP49	2010
107	How to achieve common component standardization and environmental characteristics?			
108	Investigation/case study of CBSE in Software industry.			
109	Testing tools in CBSE.	1.2.7 2.1.1		
110	Conduct a Survey with Agile and SPL experts to improve understanding.	1.2.3	SP50	2011

111	To provide support for the trade-off analysis among competing NFPs both at domain engineering and application engineering levels.	1.3	SP51	2014
112	To analyse dependencies between different kinds of NFPs and the SPL lifecycle-related practices.			
113	To understand practitioner's perceived strengths, limitations, and needs associated with using NFPs for SPL practices in the industry.			
114	Dynamically analyse systems to collect specifications and interactions at control and data flows.	1.1.1	SP52	2018
115	Early detections could benefit from structural (source code) and operational (data and control flows) analysis to evaluate the efficiency of previously detected interactions. The opposite could also be interesting, traceability from source code to models.			
116	To explore new knowledge representation models and evaluate how they could enrich the SRE (Security requirements engineering) knowledge elicitation process and, consequently, this knowledge reuse.	3.4	SP53	2015
117	Current ontology languages are limited and that there is a need for semantically richer knowledge models.			
118	At the industrial level, the question arises about the real practices of industrials on knowledge reuse during security requirements elicitations and analyses.			
119	The lack of automated support and the fact that many of the SRE methods rely on reusable knowledge that is not standard remain as issues.			
120	Authors identified 70 bad smells and 95 refactoring methods to SPL and they suggest exploring and classifying the refactoring methods listed, identifying what refactoring methods can be applied to minimize or solve some bad smells, and exploring variability smells to detect gaps in literature aims to propose new smells.	1.1.2 1.2.8	SP54	2014
121	The complex nature of variability in software product lines scenarios.	1.4.1	SP55	2017
122	The gap between problem and solution space assets.			
123	Comparison between traceability proposals for SPL and single-system development.			
124	The use of traceability to locate proper SPL assets.			
125	The lack of proper tools.	1.2.7 1.4.1		
126	Few works examined barriers of reusability, which can motivate organizations to adapt software reusability approaches.	3	SP56	2015

127	The studies about maturity models of software reuse are limited, so exploring this domain for helping organizations to audit his maturity reuse levels.			